UPPSALA
UNIVERSITET

STACKING ENSEMBLE CLASSIFICATION APPLIED TO US FLIGHT DELAY PREDICTION DURING THE COVID-19 PANDEMIC

Submitted by
Patrick Schwarz

A thesis submitted to the Department of Statistics
in partial fulfillment of the requirements
for a one-year Master of Science degree in Statistics
in the Faculty of Social Sciences

Supervisors
Yukai Yang
Per Gösta Andersson

Spring, 2022

ABSTRACT

This thesis aims to show that a Stacking Ensemble of multiple base-learners can provide a more accurate prediction of commercial flight delays between the ten largest US airports than the individual prediction models. Three types of machine learning models, namely LASSO, Random Forests and Neural Networks are used as base-learners with different hyper-parameters. A Stacking Ensemble is created by using LASSO as meta-learner. The Stacking Ensemble and the base-learners that performed best on the training data are then evaluated on a test data set. The results are compared by the metrics accuracy, ROC AUC, MCC and F1 Score. It is shown that the Stacking Ensemble is able to provide superior predictions for flight delays in comparison to the best individual models.

KEYWORDS: Stacking Ensemble, Machine Learning, Flight Delays, Classification

Contents

1. Introduction

Flight delays are costly for passengers and airlines, therefore being able to predict potential flight delays might reduce costs for both parties. It is a topic that has been researched intensively from many different angles recently (Sternberg et al., 2022; Carvalho et al, 2020).

Commercial air traffic is a large part of the economy, that has been ever growing over the last decades. This growth also led to worse on-time performance of planes due to limited resources, for example the capacity of airports (Wang et al, 2022).

However, in the recent past from 2020 to 2022, the Covid-19 pandemic caused a significant slump in flight and passenger numbers. Therefore, it is interesting to look at how flight delays can be predicted under these specific external conditions.

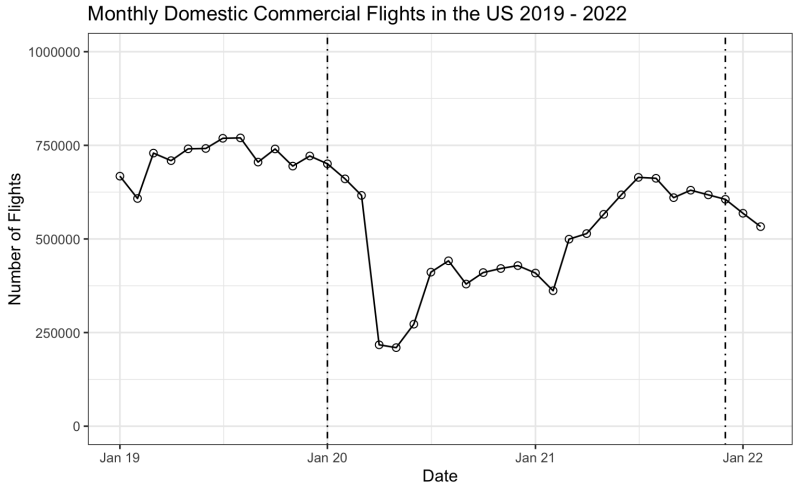

Figure 1: *Monthly domestic flights based on the data available from the Bureau of Transportation Statistics. The dotted lines mark the time frame investigated in this thesis.*

Figure 1 visualizes the extent of the impact on the flight industry in the United States during the pandemic in relationship to the year preceding the event. At the lowest point the number of commercial flights was just around one third of the year over year comparison, with flight numbers slowly increasing again over time.

The Bureau of Transportation Statistics (bts.gov) provides very detailed data about air traffic in the United States. This makes the data particularly useful to investigate. Furthermore, flight delays are more common in the US than in Europe (Campanelli et al., 2016).

As mentioned previously, flight delays are a well researched, but complex topic. A combination of many different factors could explain flight delays. One phenomenon is delay propagation, which refers to the fact that one delayed flight can cause multiple other delays with either the plane or the airport infrastructure being the bottleneck, partly due to a first-come first-served handling of flights at US airports, which supports the propagation of flight delays (Campanelli et al., 2016). Wang et al. (2020) found that the way airlines handle delays consistently predicts further flight delays and that the operational performance of airlines in this regard has improved in recent years.

Borsky and Unterberger (2019) investigated the influence of bad weather on flight delays using regressions and found a significant influence of rainfall, wind and snow on US flight performance, specifically on departure delays. Another example of influences on flight delays is the general traffic at airports, which depends on multiple factors such as time of the week, yearly seasonality and public holidays (Wang et al., 2022).

These are just some of the factors that make the prediction of flight delays a complex issue (Carvalho et al., 2021). This compelxity leads to Stacking Ensembles, which are a state-of-the-art method that proved successful at many machine learning competitions by providing the best predictions on difficult problems. Between 2005 and 2015 the number of scientific papers dealing with ensemble methods in general have tripled (Sagi & Rockach, 2018).

There are several studies about flight delay prediction using Stacking Ensembles, but they are either based on data collected before the pandemic or they have a different scope or both. For example Zhang et al.(2022) only focus on flights from a single airport or Mang and Chen (2020) who focus on a different country.

Therefore this thesis aims to answer the following research question:

Can a Stacking Ensemble predict flight delays on the 10 busiest US airports based on data collected during the pandemic better than its best base-learner models?

2. Data

The main data source for this thesis is the website of the United Stated Bureau of Transportation Statistics (www.bts.gov). They provide data for all commercial flights in the United States.

Due to computational limitations the initially very large data set was filtered down by the top ten busiest airports by flights, hence only flights between these airports are considered. The final data set consists of 624567 flights, split into 3/4 training and 1/4 testing data. The final data set consists of 60 variables, which are described in the appendix.

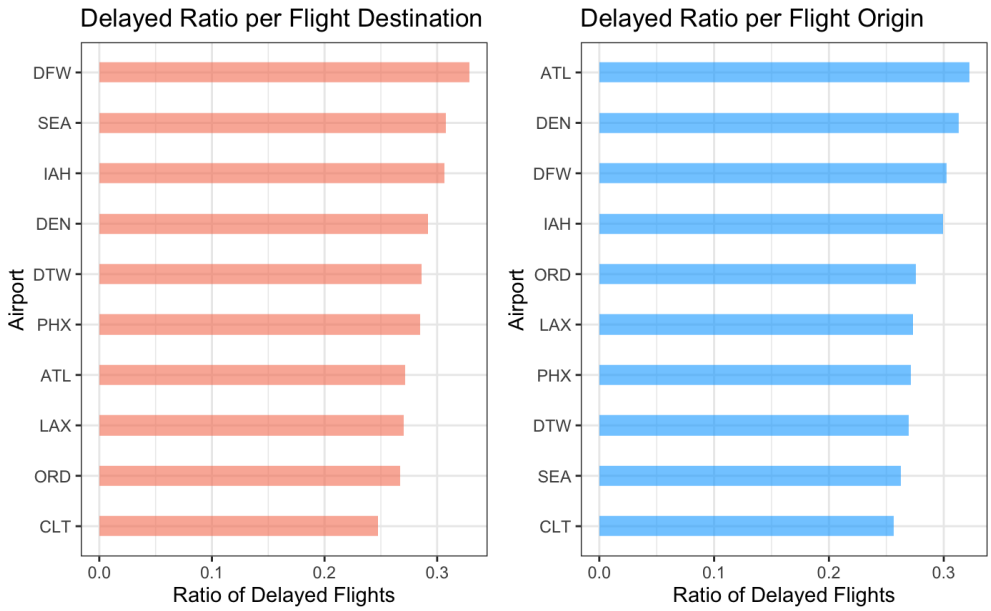

Figure 2: *Comparison of Delay Ratios for the Airports in the Data Set*

There are also notable differences between the different airports and their performance, as can be seen in figure 2. The lowest ratio of delayed flights occur at Charlotte Douglas (CLT), while Dallas Fort Worth (DFW) and Denver International (DEN) for example have higher ratios.

The initial data set includes the delay of each flight in minutes, including negative delays for flights that arrived earlier than scheduled. This was transformed into a binary response variable with the levels 'on_time' and 'delayed'. The cut off was at flights with +1 minute delayed arrival or more, which were classified as 'delayed' and the rest as 'on_time'. Note

that the response variable in the data set can be considered mildly imbalanced as can be seen from figure 3, where approximately 30% of the flights were delayed and consequently around 70% arrived by the scheduled time.

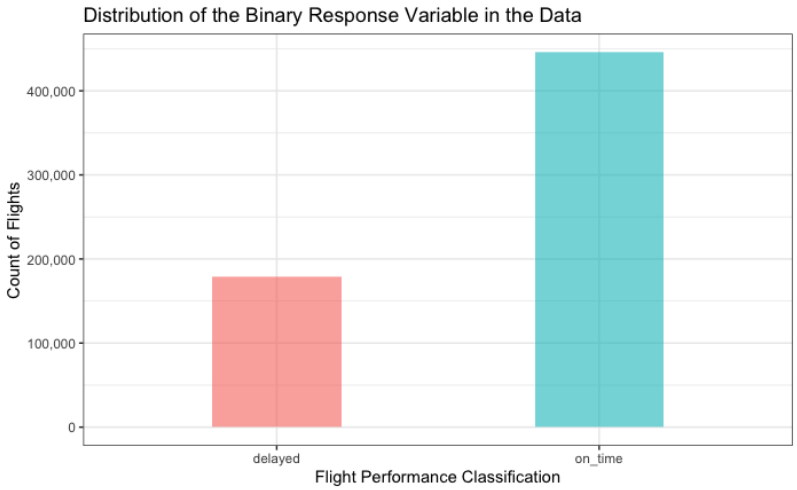

Figure 3: *Distribution of the Binary Response Variable*

By glancing at the data one can already make numerous observations that convey interesting information in regards to the relationship of some of the explanatory variables and the delay of flights. For example, figure 4 demonstrates the influence that the airlines themselves might have on the on time performance of their flights.

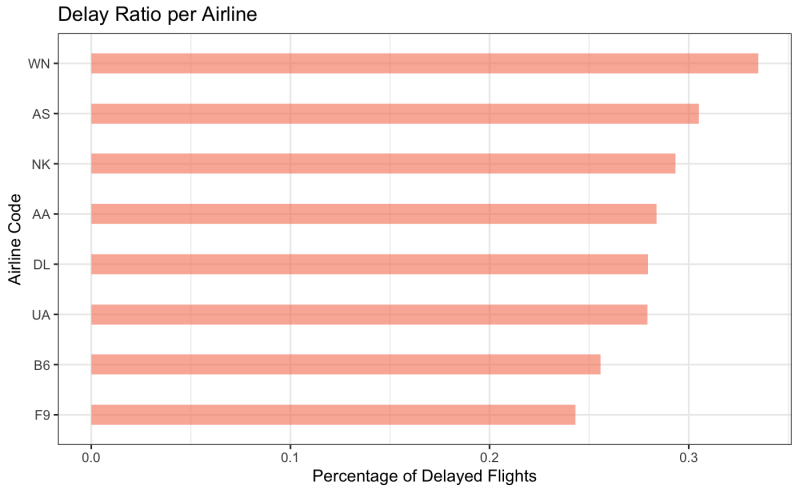

Figure 4: *Comparison of Delay Ratios for the Airlines in the Data Set*

After the download of the flight delay data from the BTS website it was augmented with information from other sources. Open source data from opensky-network.org was used to

determine the types of planes associated with each flight. The data were linked via the tail numbers that uniquely identify each airplane. Furthermore, the geographic locations for every airport was obtained from openflights.org, where the data were linked using the International Air Transport Association airport code available in both data sources. The geographic locations were then used in conjunction with the flight dates to obtain weather information by using the API of the National Oceanic and Atmospheric Administration (noaa.gov).

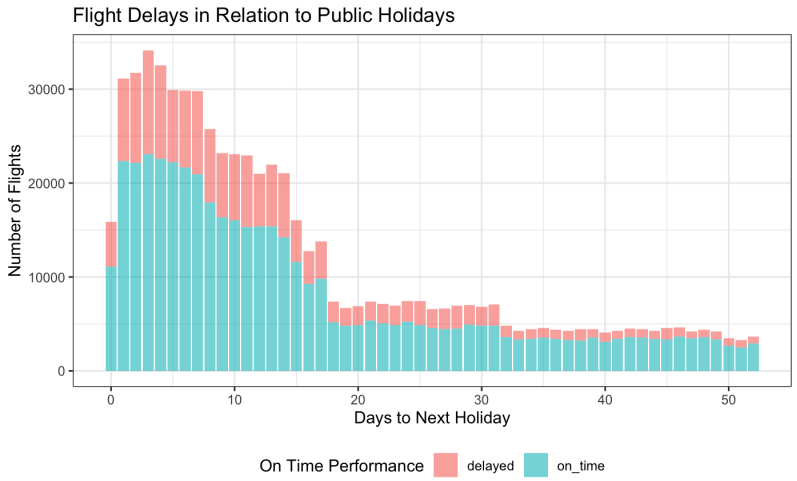

Figure 5: *Number of Flights and their Arrival Performance in Relation to the nearest Holiday*

Additionally, the distance in days to the closest national public holiday was calculated. Figure 5 shows how the number of flights increases closer in time to public holidays. Additionally it gives an indication of the quantity of delayed and on-time flights.

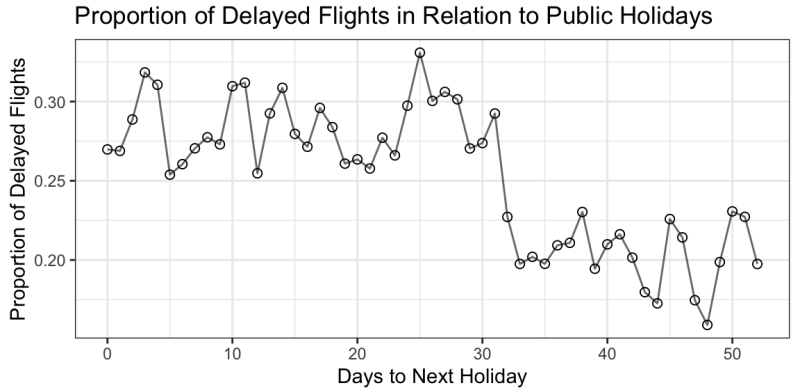

Figure 6: *Proportion of Delayed Flights in Relation to the nearest Holiday*

Figure 6 shows the relationship of the flight delay proportion in relation to the flights temporal proximity to the next US federal public holiday. There are a total of eleven federal holidays per year and they are not uniformly distributed, however it can still be seen that there are more flights close to holidays and that these flights seem to have a higher percentage of delayed flights. When looking at figure 5 and 6 combined, it appears that a larger quantity of flights per day could be related to an increasing proportion of delays.

Lastly, the flight dates were transformed using cyclical encoding to make it usable for the machine learning models.

3. Method

This section contains the theory behind the Stacking Ensemble, the base-learner models used as candidates for this thesis, their model specifications and details about how the models are evaluated.

3.1 Ensemble Learning

An integral part of ensemble learning is the combination of the learners. Dietterich (2000a) categorizes the benefits of it in three areas. Firstly, through optimizing predictions from many different starting points it is more likely to get closer to the true values as it is less likely to erroneously rely on local optima. Furthermore, by using a multitude of models the risk of overfitting on training data is reduced. Lastly, the combination of learners increases the space of solutions and reduces bias and by that increasing the chance of approximating the truth.

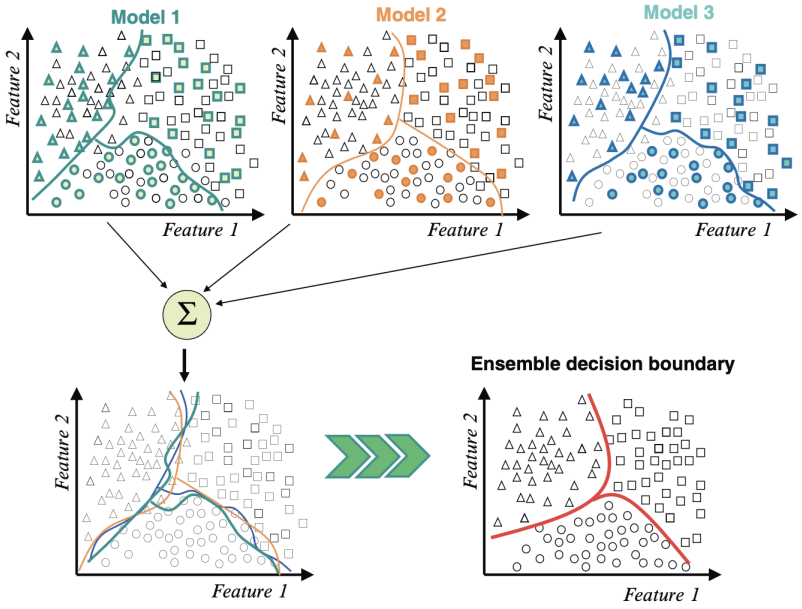

Figure 7: Visual Representation of Ensemble Classification (Polikar, 2012)

It has been shown that classification ensembles are able to predict classes better than the best individual classifiers that underpin the ensemble, early examples are for example bagging and boosting (Opitz & Maclin, 1999). Weak learners are models whose predictions are only a

limited improvement over random guesses. Even using these weak learners as base-learners in an ensemble can create strong learners with more accurate results (Zhou, 2012).

Figure 7 is a demonstration of how the principle of meta-learning functions, where it uses the results produced by base-learners and learns about how they learned from the data (Džeroski & Ženko, 2004). The figure (Polikar, 2012) depicts a classification problem with three different classes (triangle, circle and square). Each of the three models in the figure produces different decision boundaries based on two features. In this theoretical example the three models that are the base-learners have different problems in classifying the symbols correctly, for example model three works well with the circles, but has problems discriminating the triangles from the squares. The Σ-symbol represents the combination of the base-learners into the stack in the bottom left. The stack then is used as the data for the ensemble model in the bottom right, which in this example achieves perfect discrimination between the classes.

Diversity is important for ensembles, as combining base-learners that each result in different errors allows the overall errors to be reduced by using a sophisticated meta-learner (Polikar, 2006).

Ensembles can be distinguished by being heterogeneous or homogeneous. Homogeneous ensembles only use different variations of one single type of learners where diversity can be for example achieved through their respective hyper-parameters. In contrary heterogeneous ensembles combine multiple different types of learners (Bian & Wang, 2007).

While often times choosing a simple model is preferred, the relatively complex combination of many diverse models is very likely to improve the overall generalization capability (Elder, 2018).

For classification the simplest approach is using fixed combining rules, such as majority voting or averaging (Kittler, 1998), but many more options to combine multiple models exist. The one this thesis is focusing on is the Stacking Ensemble approach.

3.2 Stacking Ensemble

Originally introduced by Wolpert (1992) Stacking Ensembles, or as he called the procedure Stacked Generalization, aim to improve generalization accuracy. The primary way of achieving this is to combine multiple base-learners. This procedure is described by Wolpert as an advanced form of cross-validation. The combination procedure in this case is called a meta-learner.

Ting and Witten (1999) identify the choice of meta-learner and the composition of the data set D' that is used for stacking as significant issues for Stacking Ensembles.

Unlike other methods such as bagging, meta-learners do not use different variations of subsets of the training data, but instead create their own transformation of the data based on the predictions that are made by the different base-learners (Vilalta & Drissi, 2002).

For classification using the class probabilities that each base-learner assigned for each observation as the meta-learner data set has shown to provide better results than using the predicted class (Ting & Witten, 1999).

Stacking has been found to improve prediction quality, while other methods are better suited to optimize other outcomes, such as bagging for decreasing variance (Nti et al., 2020; Zhou, 2012).

The base-Learners h_t create predictions through the algorithms $\mathcal{L}_1, \ldots, \mathcal{L}_T$ based on the training data D. The probability for one class and the true outcome are combined to create the data set D'. While this is the procedure used in this thesis, there are other ways of creating D', for example by also including the initial training data D. The meta-learner h' is trained with the algorithm $\mathcal{L}$ on D' to create the new Stacking Ensemble $H(\boldsymbol{x})$ that can be used to predict on unseen test data (Zhou, 2012). The process is summarized in algorithm 1.

Many different algorithms can be used with a classification meta-learner in a Stacking Ensemble, for example XGBoost, Logistic Regression, Random Forest, LASSO or Neural Networks.

Algorithm 1: Stacking Ensemble Algorithm (Zhou, 2012)

Input : Data set D = $\{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), ..., (\boldsymbol{x_m}, y_m)\}$

Base-Learners $\mathcal{L}_1, \ldots, \mathcal{L}_T$

Meta-Learner $\mathcal{L}$

Output: $H(\boldsymbol{x})$=$h'\left(h_1(\boldsymbol{x}), \ldots, h_T(\boldsymbol{x})\right)$, where $h_t(\boldsymbol{x})$ denotes the output of the t-th base-learner with $\boldsymbol{x}$ as a vector of class probabilities for all observations. These are combined by the meta-learning algorithm h' into the combined class probability $H(\boldsymbol{x})$.

for $t = 1, ..., T$: **do**
  $h_t = \mathcal{L}_t(D)$
end
Generate a new data set $D' = \emptyset$
for $i = 1, ..., m$: **do**
  for $t = 1, ..., T$: **do**
    $z_{it} = h_t\left(\boldsymbol{x_i}\right)$, where the t-th learner is applied to the i-th observation in the input vector $\boldsymbol{x}$, resulting in the it-th class probability z_{it}, which is the output of base-learner h_t on $\boldsymbol{x_i}$.
  end
  Combine the base-learner class probabilities into the new data set: $D' = \{(z_{i1}, \ldots, z_{it}, y_i)\}_{i=1}^{m}$.
end
Train the meta-Learner h' by applying the Meta-Learner algorithm $\mathcal{L}$ to the new training data set for D'.
$h' = \mathfrak{L}\left(D'\right)$

3.3 Learners

3.3.1 Logistic LASSO

LASSO is an acronym for least absolute shrinkage and selection operator. This statistical method's goal is coefficient shrinkage and automatic selection of the predictor variables. The

logistic variant is essentially a logistic regression model with a penalty term added that regularizes the coefficients. Due to the variable selection properties the LASSO is well suited for data sets with $n >> p$ and due to the sparsity induced by the method it can handle large data sets. (Tibshirani, 2011)

A categorical response variable is estimated using one or multiple predictor variables. In logistic regression, predictor variables may be categorical or numerical. A sigmoid function that limits outputs between 0 and 1 is used to convert the relationships between the predictor variables into a binary categorical estimate, for which the numerical value conveniently is the equivalent probability for the class membership of each observation. The logistic function, for simplicity here exemplified with a single predictor, is represented in (1) based on James(et al., 2012), with

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}, \tag{1}$$

where p(x) can take any value between 0 and 1. This can be transformed into the odds ratio in (2) by using cumulative logistic distribution.

$$\frac{p(X)}{1 - p(X)} = e^{\beta_0+\beta_1 X} \tag{2}$$

The odds after the application of the natural log give the log odds as shown in (3):

$$\log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X \tag{3}$$

Based on the probabilistic nature of the logistic regression one can make use of the likelihood estimation in which (4) can be used to compute the joint probabilities under the assumption that $P(y_i = 1) = p$ and $P(y_i = 0) = 1 - p$.

$$L(\beta_0, \beta) = \prod_{i=1}^{n} p(x_i)^{y_i} (1 - p(x_i))^{1-y_i} \tag{4}$$

Due to limitations of calculations with the likelihood function and its products, the negative log of the likelihood function is used instead, as shown in (5), where the negative log loss function is to be minimized:

$$L_{log} = \ell(\beta_0, \beta) = \sum_{i=1}^{n} -\log 1 + e^{\beta_0 + \beta x_i} + \sum_{i=1}^{n} y_i (\beta_0 + \beta x_i) \tag{5}$$

Logically speaking, the values for $\hat{\beta}_0$ and $\hat{\beta}_1$ are optimized in a way that if they are entered in (5) as many observations as possible are associated with their true class.

Tibshirani (1996) developed the idea of the $L_1 - penalty$ to reduce the errors in OLS estimation, but it can similarly be applied in the logistic regression context. The effect of the penalty is shrinkage of the coefficients, which is based on the absolute magnitude of the coefficients. Consequently, coefficients of a LASSO model can be shrunk to 0 and by that remove the respective predictor from the estimation. The LASSO has a hyper-parameter λ that is multiplied by the penalty and thereby establishes the magnitude of the penalty. This is represented in (6), with

$$L_1 - penalty = P(\beta) = \sum_{j=1}^{p} |\beta_j|, \tag{6}$$

where λ can take any value between 0 and ∞. The penalizing term multiplied by λ is then simply added to the log-likelihood formula in equation (7):

$$L_{log} + \lambda \sum_{j=1}^{p} |\beta_j| \tag{7}$$

From this can be deduced that if $\lambda = 0$ the LASSO becomes a regular logistic regression without any shrinkage applied to its coefficients. The value for λ is in practice determined by a cross-validation procedure (Harrel, 2015).

The LASSO method is represented in algorithm 2:

Algorithm 2: Logistic LASSO Algorithm (Van Loon et al., 2010)

Input : A set of features X

Binary outcomes y

Output: A learned function $\hat{f}$

Define a set of candidate tuning parameter values Λ.

Randomly split the data into K cross-validation folds of roughly equal size.

foreach $\lambda \in \Lambda$ **do**

  for $k = 1$ *to* K **do**

    Optimize the penalized likelihood using only the observations outside of fold k.

    Apply the trained model to obtain an error for each observation on fold k.

  end

  Average the cross-validation error across all observations.

end

Choose λ^* to be the value in Λ with the lowest cross-validation error.

Optimize the penalized likelihood using all observations and the penalty parameter vaule to obtain a set of parameters $\hat{\beta}_0^{(\lambda^*)}$, $\hat{\beta}^{(\lambda^*)}$.

Return the learned function $\hat{f}(X) = 1/\left(1 + \exp\left(-\hat{\beta}_0^{(\lambda^*)} - X\hat{\beta}^{(\lambda^*)}\right)\right)$.

3.3.2 Random Forest

A Random Forest is an ensemble of Decision Trees with the trees as the base-learners and the Random Forest as the meta-learner. Random Forests can either be constructed from regression or classification trees. The disadvantage of Decision Trees is their high variance, which can be overcome by the combination of many trees into a forest (James et al., 2021).

Simply using Decision Trees also tends to result in overfitted models that are not working well on new data. This is where Random Forests a come into play, as they can be used to improve prediction accuracy on unseen data compared to a Decision Tree due to the Law of Large Numbers (Breiman, 2001).

In order to de-correlate the Decision trees from each other, the predictor variables in each tree are a randomly selected subset m of the entirety of predictors p. This is done so the

trees of the model are not focusing on few strong predictors where they would end up very similar to each other. Therefore, by increasing the number of trees B the accuracy of the classifier improves and overfitting is prevented. The minimum amount of nodes per tree is n_{min} (James et al., 2021; Breiman, 2001).

Bagging is used where repeated bootstrap samples Z^{*B} are taken from the data set D with replacement and of size N. The classifier is then trained using this sample to obtain the model $\hat{f}^{*b}(x)$. The procedure is repeated B times, after which uniform aggregation leads to (8):

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x) \tag{8}$$

This results in better performance where the $avgErr(\hat{f}^b(x)) \geq Err(avg(\hat{f}^b(x))$. The bagging produces B trees, after which a majority vote is cast between the trees to determine class C of observation x.The Random Forest procedure is summarized in algorithm 3:

Algorithm 3: Random Forest Classification Algorithm (Hastie et al., 2009)

Input : Training Data D

Parameters $\{B, n_{min}, m, p, N\}$

Output: Tree Ensemble

Class Prediction

for $b = 1$ *to* B **do**

(a) Draw a bootstrap sample Z^* of size N from D.

(b) Grow a Random Forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree until n_{min} is reached .

i. Select m variables at random from p.

ii. Pick the best variable/split-point among m.

iii. Split the node into two daughter nodes.

end

Output the ensemble of tree $\{T_b\}_1^B$

Let $\hat{C}_b(x)$ be the class prediction of the b-th tree.

Then $\hat{C}_{\text{rf}}^B(x) =$ majority vote $\left\{\hat{C}_b(x)\right\}_1^B$.

3.3.3 Neural Network

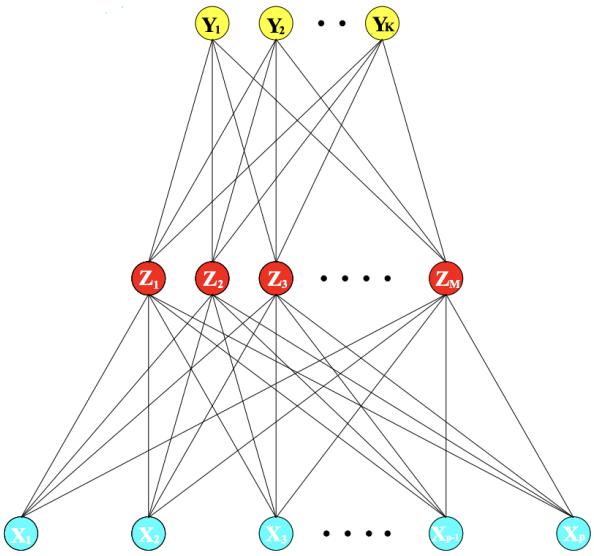

Figure 8: *Shallow Feed-Forward Neural Network (Hastie et al., 2009)*

Neural Networks can be distinguished by the amount of hidden layers in the network. In this paper the focus is on single layer neural networks, which means that the structure is similar to figure 8, with a network consisting of an input layer, a single hidden layer and an output layer. The input layer of a classification model consists of nodes based on the predictor variables X_1 to X_p. The input layer nodes feed into each of the M nodes of the hidden layer. From there, the hidden layer nodes feed into the output layer with its K nodes, which represent each of the classes predicted. Neural Networks can include non-linear transformations and hence are well suited for non-linear relationships within the data (Hastie et al., 2009).

Each of the M nodes in the hidden layer has an activation Z, with equation (9)

$$Z_m = h_m(X) = g\left(w_{m0} + \sum_{j=1}^{p} w_{mj} X_j\right), \tag{9}$$

where g is the non-linear activation function, $h_m(X)$ a transformation of the input and w the weights (James et al., 2021).

Common activation functions are ReLU, linear, sigmoid and softmax. For this study only the softmax function (10) is of relevant. It formulated as

$$g(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)}, \tag{10}$$

where g is the function and x the input. The denominator normalizes the result of the function to a value between 0 and 1, which similar to the sigmoid in logistic regression can be interpreted as probabilities (James et al., 2021).

The fitting of Neural Network is concerned with the weights that represent the unknown parameters in the model, where θ is the set of all weights. The components of θ are shown in (11) (Hastie et al., 2009):

$$\{\alpha_{0m}, \alpha_m; m = 1, 2, \ldots, M\} \quad M(p{+}1)weights, \{\beta_{0k}, \beta_k; k = 1, 2, \ldots, K\} \quad K(M{+}1)weights. \tag{11}$$

The fit of a neural network used for classification can be measured by the cross-entropy (12) that is calculated with

$$R(\theta) = -\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik} \log f_k\left(x_i\right), \tag{12}$$

where the objective function $R(\theta)$ is minimized. A common procedure for the minimization in stochastic gradient descent, but computationally more efficient options exist, such as the ADAM optimizer. The combination of cross-entropy loss and softmax activation is the equivalent of logistic regression within the nodes of the hidden layer (Hastie et al., 2009; Kingma & Ba, 2015).

Regularization can be used in neural networks to address the overfitting that $R(\theta)$ likely inhibits (Hastie et al., 2009). While there are multiple options, dropout regularization is used for the neural network in this study. The technique is based on randomly dropping nodes from the input and hidden layer of a single layer neural network to reduce overfitting. A new subset is dropped for every training sample. The constant ϕ determines the proportion of dropped nodes. Because the overall number of nodes used in each iteration this neural network is lower, the weights are scaled using the factor $1/(1-\phi)$ (James et al., 2021).

3.4 Model

Figure 9 visualizes the definition of the different candidate models that were used to create the Stacking Ensemble. Different models of base-learners were configured with different hyper-parameters or combinations thereof. Due to computational constraints the amount of models and hyper-parameter combinations was limited.

The four LASSO models differ only in their λ value, the different values that were used are 0, 0.0001, 0.001 and 0.01. The predictor variables were one-hot encoded and normalized.

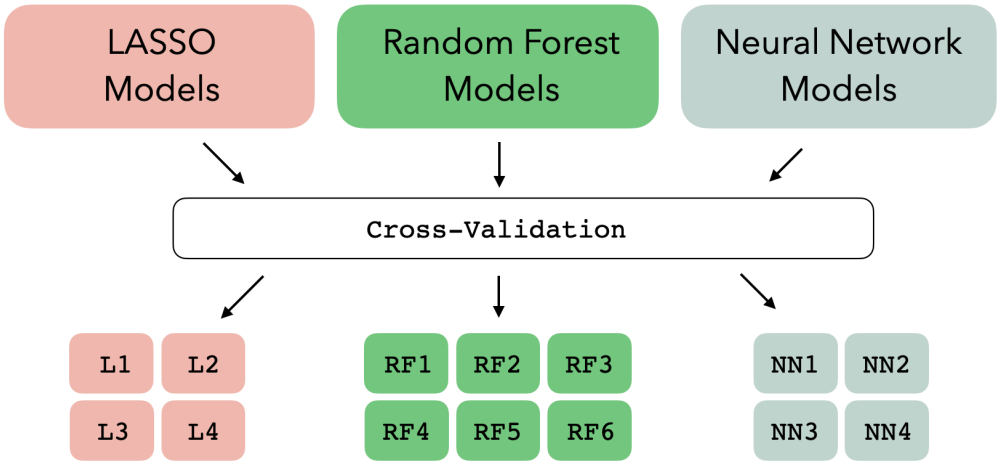

Figure 9: *Model Definitions for the different Base-Learners that became Candidate Models for the Stacking Ensemble after Cross-Validation, inspired by stacks.tidymodels.org*

For the Random Forest models the number of trees and the number of randomly selected predictors was varied between the models

The four Neural Network models had different numbers of hidden layer node with everything else equal.The predictor variables were one-hot encoded and normalized. Dropout was set to 10%, batch size to 32. The minimization of the binary cross-entropy was done by using the ADAM optimizer and the activation function was softmax.

The results of the base-learner models, specifically the class probabilities, and the true outcomes from the data set were combined into the data stack as represented in figure 10.

Figure 10: *Representation of the Data Stack with the Results of the Base-Learners that was used for Training the Stacking Ensemble*

The LASSO meta-learner used 25 bootstrap re-samples of the data stack with λ values in the range of 10^{-6} to 10^{-1}. This lead to the creation of the combined Stacking Ensemble model, which was evaluated on the test data.

The best base-learner models per type were determined from the test data by their ROC

AUC and were then also evaluated on the test data to have a benchmark for the Stacking Ensemble.

3.5 Model Evaluation

This section describes by which metrics the models are evaluated and why they are evaluated. The first applications of model evaluation are the candidate base-learner models for the ensemble, here it is determined to which extent the base-learners are included in the final Stacking Ensemble. This is purely based on test data and only ROC AUC is used, as there is a limitation within the R package for this procedure.

After the ensemble composition is determined, multiple metrics are used to evaluate the Stacking Ensemble and the best performing base-learners on the test data and against each other. The intuition is based on figure 7 in section 3.1, where it is implied that a Stacking Ensemble should be performing better at classification than its base-Learners. To reduce the number of models compared against the ensemble, the base-learner models are first evaluated on the training data to find the best one for each type.

3.5.1 Confusion Matrix

One way to summarize to performance of a model is the confusion matrix (Table 1), which displays the information about how many observations of each class are correctly or falsely classified by the model.

Table 1: Confusion Matrix

	Actual True	Actual False
Predicted Positive	True Positive (TP)	False Positive (FP)
Predicted Negative	False Negative (FN)	True Negative (TN)

The values that are represented within the confusion matrix can be used as building blocks for the other metrics.

3.5.2 Accuracy

Accuracy is simply the rate of correctly classified observations calculated as follows (13):

$$\text{ACC} = \frac{1}{n}\sum_{i=1}^{n} I\left(y_i \neq \hat{y}_i\right) = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{13}$$

Accuracy can be a misleading metric in cases of imbalanced data sets, for example in cases where the classifier only guesses the majority class (Chicco & Jurman, 2020).

3.5.3 ROC AUC

The Receiver Operator Characteristic (ROC) graphically represents the relationship between the false positive rate or specificity (14) and the true positive rate or sensitivity (15) for binary classification tasks and can be used to diagnose the fit of a model (Kuhn & Johnson, 2013).

$$Specificity = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{14}$$

$$Sensitivity = Recall = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{15}$$

The performance of a classifier can be assessed by determining the Area Under the Curve (AUC) of the ROC using equation (16).

$$ROCAUC = \frac{1 + \text{TP} - \text{FP}}{2} \tag{16}$$

According to Huang and Ling (2005) the AUC has advantages over accuracy when it comes to consistency and discriminancy. The ability to discriminate between the classes can be interpreted by the following heuristic based on Hosmer et al. (2013):

$$IF \begin{cases} ROCAUC = 0.5 & \text{No Discrimination} \\ ROCAUC < 0.7 & \text{Poor Discrimination} \\ 0.7 \leq ROCAUC < 0.8 & \text{Acceptable Discrimination} \\ 0.8 \leq ROCAUC < 0.9 & \text{Excellent Discrimination} \\ ROCAUC \geq 0.9 & \text{Outstanding Discrimination} \end{cases}$$

3.5.4 F1 Score

The F1 Score (18) combines Precision (17) and Recall (15) into a measure for the entire model and it increases if both values are balanced, which makes it particularly useful when both classes are of equal importance (Chinchor, 1992). This behavior can at the same time be problematic in imbalanced data sets (Chicco & Jurman, 2020).

$$\text{Precision } = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{17}$$

$$\text{F1 Score } = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}} = 2 \cdot \frac{\text{precision } \cdot \text{ recall}}{\text{precision } + \text{ recall}} \tag{18}$$

3.5.5 MCC

Matthews Correlation Coefficient (MCC) has the advantage to be indifferent to imbalanced data sets and class swapping (Chicco & Jurman, 2020).

$$\text{MCC} = \frac{\text{TP} \cdot \text{TN} - \text{FP} \cdot \text{FN}}{\sqrt{(\text{TP} + \text{FP}) \cdot (\text{TP} + \text{FN}) \cdot (\text{TN} + \text{FP}) \cdot (\text{TN} + \text{FN})}} \tag{19}$$

The values of the MCC can range between -1 and 1, where -1 represents perfect misclassification and 1 perfect classification (Chicco & Jurman, 2020).

4. Results

This section presents the results of three different procedures. First the the results of the base-learner models on the training data set set will be demonstrated. This is followed by the tuning of the LASSO meta-learner and its results in order to compose the Stacking Ensemble. Lastly, the performance of the Stacking Ensemble is assessed based on the test data set and then compared against the performance of the best LASSO, Random Forest and Neural Network models. Note that for all of the the tables in section 4.1 the ROC AUC values are the means of the N-fold cross validation procedure. For all models five folds are used. The standard error is a mean based on the results of the different folds.

4.1 Base-Learner Results on Training Data

The base-learner results are presented in table form. They are relevant because they determine which models are the benchmarks for the stack. To decide which base-learner has the best performance only the ROC AUC metric is used, which is captured as the mean value across the five folds of the cross-validation procedure. Within the tables the models are ordered descending by their ROC AUC value.

Table 2: Performance of the LASSO Base-Learner Models

Model	λ	**Metric**	**Mean**	**N**	**Std_Err**
L1	0	ROC AUC	0.63876	5	0.00034
L2	0.0001	ROC AUC	0.63876	5	0.00034
L3	0.001	ROC AUC	0.63776	5	0.00033
L4	0.01	ROC AUC	0.61495	5	0.00072

Table 2 shows the performance of the LASSO models. It is notable that the model without penalty and the model with $\lambda = 0.0001$ perform exactly the same. Therefore the simpler model, in this case the model that is equivalent to a logistic regression model is determined to be the best model.

Table 3: Performance of the Random Forest Base-Learner Models

Model	n_{min}	Trees	Metric	Mean	N	Std_Err
RF5	8	500	ROC AUC	0.73919	5	0.00071
RF6	16	500	ROC AUC	0.73849	5	0.00081
RF3	8	250	ROC AUC	0.73783	5	0.00062
RF4	16	250	ROC AUC	0.73733	5	0.00069
RF1	8	25	ROC AUC	0.71995	5	0.00054
RF2	16	25	ROC AUC	0.71974	5	0.00071

Table 3 shows the results of the training procedure for the Random Forest base-learners configurations. Differentiating are the randomly selected predictors for each tree and the number of trees in the forests. The minimum node size was left static for this procedure with a value of 15. As can bee seen from the results, the number of randomly selected predictors was not very influential for the performance of the individual models. In contrast the number of decision trees for the random forests had a more significant impact on the ROC AUC of the models. The best Random Forest configuration uses 500 trees and 8 randomly selected predictors.

Table 4: Performance of the Neural Network Base-Learner Models

Model	Hidden Units	Metric	Mean	N	Std_Err
NN4	250	ROC AUC	0.70293	5	0.00068
NN2	150	ROC AUC	0.70279	5	0.00119
NN3	200	ROC AUC	0.70248	5	0.00049
NN1	100	ROC AUC	0.70206	5	0.00071

Table 4 shows the the best performing Neural Network model was the one with the most hidden units. The general trend appears to be that more nodes improve the metric. However, there is some inconsistency for the model with 150 nodes. The cross-validation for this model

has a higher standard error than the other four Neural Networks and it is also higher than for any other base-learner model. Interestingly, when comparing ROC AUC and accuracy of the Neural Networks it can be observed in figure 11 that at 150 nodes the accuracy dips strongly and is better at lower and higher node counts.

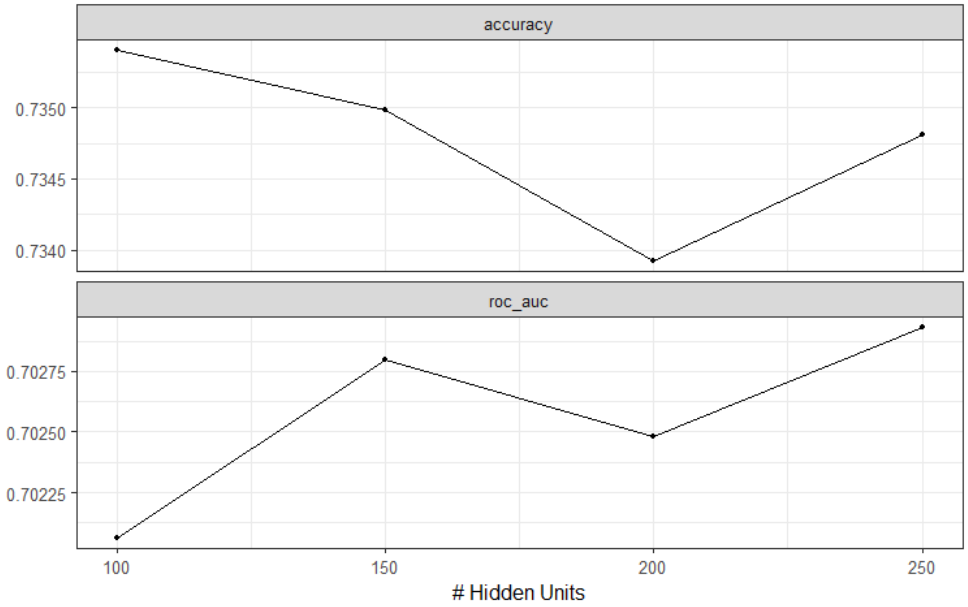

Figure 11: *Comparison of Accuracy and ROC AUC for the Neural Networks during Training*

Figure 12 compares the performance of all base-learners visually. For the accuracy metric all three types of models are clustered relatively distinct from each other with small performance differences within the different groups. One observation is that the the two weakest of the Random Forest models are slightly worse than the others. The performance variation between the Neural Network modes is slightly declining, while for the Lasso models the last ones drops of strongly compared to the other three. Generally the Random Forest models outperform the other two types, while the Neural Networks beat the Lasso models.

The order of models is similar for the other metric. The graph for the ROC AUC shows larger differences between the types as well as within the the types. The two weaker Random Forest models fall off more for this metric and between the Neural Networks and the other Random Forest models. The LASSO model look much worse relatively to the other model types and the weakest model falls off more for the ROC AUC than for accuracy.

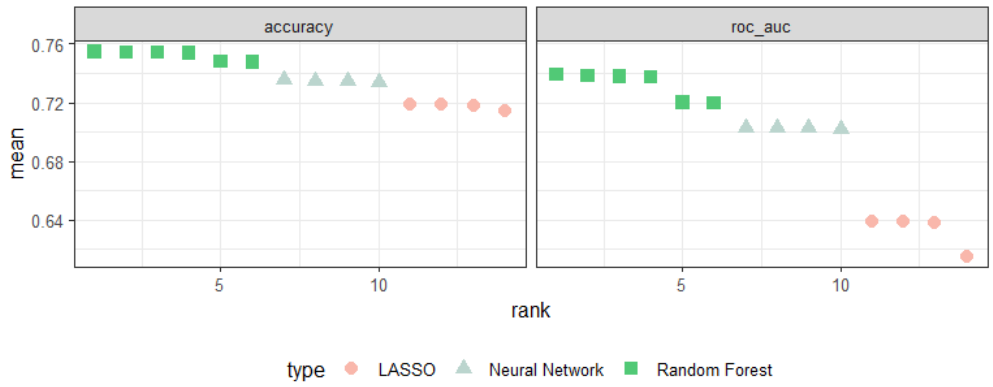

Figure 12: *Comparison of Accuracy and ROC AUC for the Base-Learners during Training*

4.2 Generation of the Stacking Ensemble

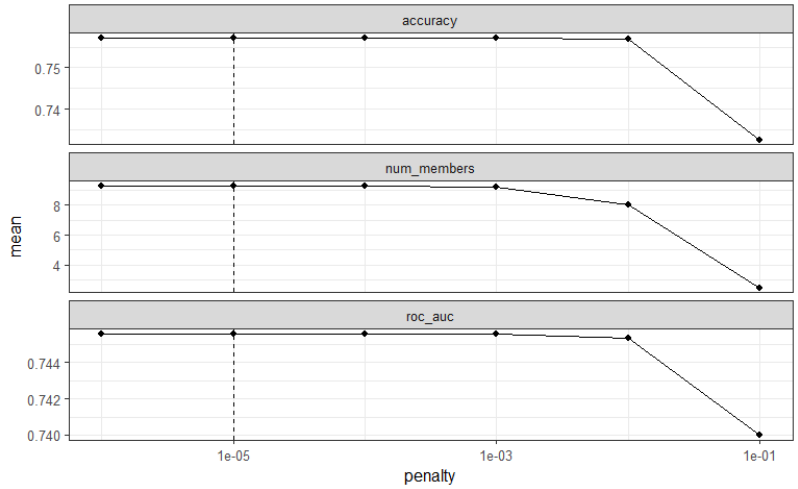

Figure 13: *Training Performance of the Stacking Ensemble on the Data Stack*

Figure 13 above shows the result of the tuning of the Stacking Ensemble model based on the training data results of the base-learner models. The graph sets the number of base models in the Stacking Ensemble into related to the accuracy and the ROC AUC values achieved with the different configurations fed into the Lasso meta-learner model. Due to the similarity off some of the base-learner predictions the regularization of the Lasso removes some of the models. From a total of 14 models the lowest level of regularization at $\lambda = 0.000001$ sets the

coefficients of three models to zero as part of the automatic feature selection properties in the Lasso. When the amount of base-learners gets reduced further the performance of the Stacking Ensemble drops of for accuracy as well as ROC AUC. The tuning process determined the optimal amount of regularization to be at $\lambda = 0.00001$ as marked with the dotted line in the graph.

Table 5: Models Included in the Stacking Ensemble with Coefficients

Stacking Ensemble Member	**Coefficient**
Random Forest 5	1.6806
Random Forest 6	1.5868
Random Forest 4	0.7188
Random Forest 3	0.5399
Neural Network 4	0.3581
Neural Network 3	0.2959
Neural Network 2	0.2762
Neural Network 1	0.2220
Random Forest 2	0.0257
Random Forest 1	0.0141

The models are not necessarily in order of their ROC AUC results. Random Forest models 5 and 6 have by far the biggest coefficients. Notably is also that all Neural Networks have similar coefficients. The magnitude of the coefficients is likely driven by the difference in information the base-learners convey to the ensemble with their predictions.

4.3 Evaluation on Test Data

Figure 14 shows the confusion matrices for the best base-learners of each type and the stacking ensemble after the evaluation on the test data set.

The confusion matrix for the Stacking Ensemble based on the evaluation on the test data in figure 14 shows that the true values for the test data had 71.4% of flights on time and 28.6%

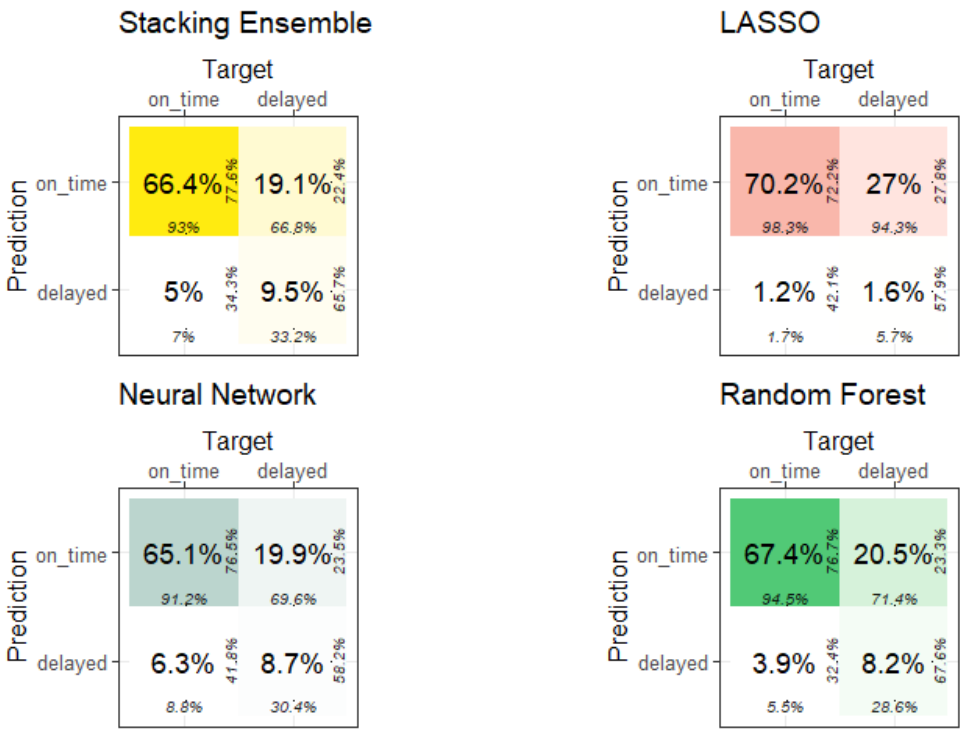

Figure 14: *Confusion Matrices for the Stacking Ensemble and the best Base-Learners*

delayed. In contrast to that the model predicted 85.5% of the flights to be on time and 14.5% to be delayed. With these predictions 93% of the on time flights were correctly predicted while 7% of the on time flights were falsely predicted to be delayed. Of the truly delayed flights only 33.2% of flights were correctly predicted and 66.8% were wrongly classified as delayed.

The small percentages in the figure indicate the vertical and horizontal class proportions for the true and predicted values. For example, the ensemble predicted 93% of the on-time flights correctly, while 7% were wrongly classified as delayed. 77.6% of the as on-time predicted flights by the ensemble were actually on-time, while 22.4% were in truth delayed.

It can be seen that the LASSO model is the best at correctly classifying the majority group, but is really unsuccessful when it comes to the minority class. Hence this model has problems dealing with the imbalance of the dependent variable, while at the same time the Stacking Ensemble has the highest percentage of correct predictions on the minority class of all models, but is only third best on the majority class.

Table 6: Evaluation Metrics based on Test Data

Model	**Accuracy**	**ROC AUC**	**MCC**	**F1**
Stacking Ensemble	0.75889	0.75122	0.33683	0.84631
Random Forest	0.75626	0.74409	0.32009	0.40192
Neural Network	0.73796	0.71079	0.27372	0.39888
LASSO	0.71802	0.63827	0.11015	0.10351

Table 6 shows that by every single metric the Stacking Ensemble is superior to all other models. The models are ranked the same for every single metric, which allows for an easy ranking between them. The general types of the base-learners perform similar in relation to each other as during the training phase, while the Stacking Ensemble outperforms the best individual models.

While all models are relatively close to each other in accuracy with values between 71.8% and 75.95% the differences in the other metrics are more pronounced, even if the metrics do not cover the same ranges. The LASSO falls off a bit more on the ROC AUC metric and even more for the MCC and the F1 Score, which indicates that a linear model is not well suited for this task. While the Random Forest is close to the performance of the Stacking Ensemble in accuracy, ROC AUC and MCC it performs relatively worse on the F1 Score. It can be said that overall the Stacking Ensemble predicts flight delays the best of the models evaluated.

Figure 15 shows the Receiver Operator Characteristics for the Stacking Ensemble and the best performing models of each type of base-learner based on the test data set. The dotted line represents the performance of flipping coins. It can be observed that the Lasso models clearly performs the worst. The Neural Network comes close to the two best models but is still clearly outperformed. The curves for the Stacking Ensemble and the Random Forest model are very close performance wise and even overlap each other. From the ROC AUC values it is visible that the value for the Stacking Ensemble is superior. The graph shows that the Random Forest model performs slightly better between x-axis values 0 and 0.25, but is outperformed by the Stacking Ensemble between 0.25 and 1, which overall leads to

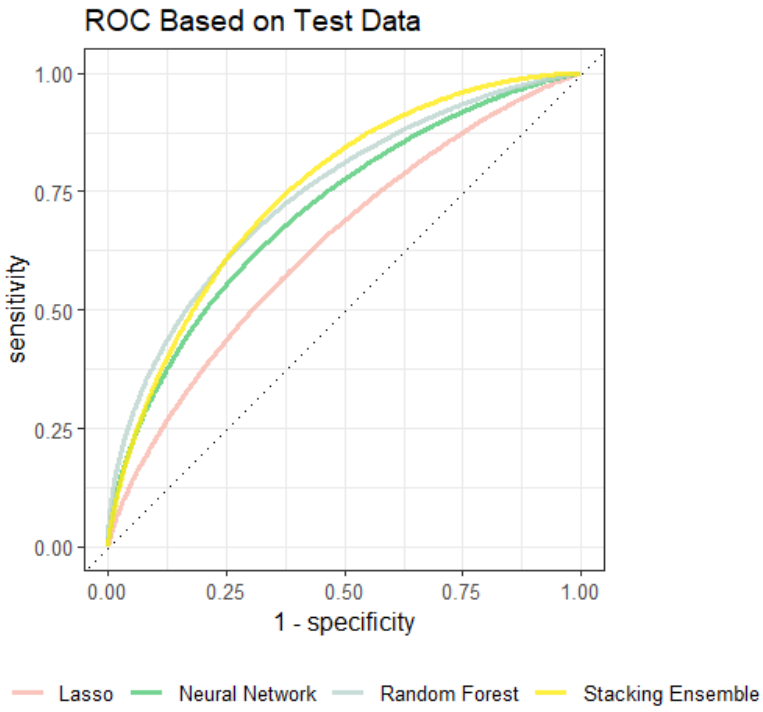

Figure 15: *The ROCs for the different Models visually indicate the respective AUC*

the conclusion that even visually, the Stacking Ensemble is the best model according to this metric based on the evaluation on the test data set.

5. Conclusion and Discussion

Like multiple studies before, performance of predictions based on unseen data has been improved by using a Stacking Ensemble approach. The Stacking Ensemble model outperforms every of its constituent base-learners in all metrics that were used for evaluation. It can therefore be confirmed that a Stacking Ensemble can predict flight delays on the 10 busiest US airports based on data collected during the pandemic better than its best base-learner models.Interestingly, the best performing base-learner is also an ensemble model, showing that the concept of ensembles in general can be very powerful if the extra computational cost is no issue.

The combination of the base-learners shows that not only the best performing base-learners are combined with descending magnitude of meta-learner coefficients. Instead some weaker models have bigger coefficients than models that were performing better themselves, which indicates that these models convey some additional useful information into the meta-learner.

This however does not mean there is no room for further improvement of the predictions on this type of data. Especially since the RIC AUC for the Stacking Ensemble is only 0.751, which can be only classified as fair discrimination. The model that is used by Zhang et al. (2021) achieves a better values for ROC AUC, accuracy and F1 score, which are the commonly used metrics. It has to be mentioned that they use more diverse models and have a larger sample size. That means there is still significant potential to increase the predictions of flight delays.

There are many paths for further investigations and improvements. Firstly, the number of models for each base-learner type could be increased and more variety in their hyper-parameters could be utilized. Secondly, additional types of base-learners could be used, for example Support Vector Machines or XGBoost models. Thirdly, a different meta-learner model could be used, such as a Random Forest or Neural Network. Fourthly, one could try to take different approaches to combining the data stack, for example by including the initial data. Lastly, the scope of the data could be extended, for example by including more airports, a longer time period or additional variables to capture other potential causes for flight delays.

6. References

Bian, S., & Wang, W. (2007). On diversity and accuracy of homogeneous and heterogeneous ensembles. *International Journal of Hybrid Intelligent Systems*, *4*(2), 103–128.

Borsky, S., & Unterberger, C. (2019). Bad weather and flight delays: The impact of sudden and slow onset weather events. *Economics of Transportation*, *18*, 10–26.

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32.

Campanelli, B., Fleurquin, P., Arranz, A., Etxebarria, I., Ciruelos, C., Eguíluz, V. M., & Ramasco, J. J. (2016). Comparing the modeling of delay propagation in the US and European air traffic networks. *Journal of Air Transport Management*, *56*, 12–18.

Carvalho, L., Sternberg, A., Maia Gonçalves, L., Beatriz Cruz, A., Soares, J. A., Brandão, D., Carvalho, D., & Ogasawara, E. (2021). On the relevance of data science for flight delay research: A systematic review. *Transport Reviews*, *41*(4), 499–528.

Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, *21*(1), 6.

Chinchor, N. (1992). MUC-4 evaluation metrics. *Proceedings of the 4th Conference on Message Understanding - MUC4 '92*, 22.

Couronné, R., Probst, P., & Boulesteix, A.-L. (2018). Random forest versus logistic regression: A large-scale benchmark experiment. *BMC Bioinformatics*, *1*, 1–14.

Dietterich, T. G. (2000). Ensemble Methods in Machine Learning. In G. Goos, J. Hartmanis, & J. van Leeuwen (Eds.), *Multiple Classifier Systems* (Vol. 1857, pp. 1–15). Springer Berlin Heidelberg.

Džeroski, S., & Ženko, B. (2004). Is Combining Classifiers with Stacking Better than Selecting the Best One? *Machine Learning*, *54*(3), 255–273.

Elder, J. (2018). The Apparent Paradox of Complexity in Ensemble Modeling *. In *Handbook of Statistical Analysis and Data Mining Applications* (pp. 705–718). Elsevier.

Harrell, F. E. (2015). *Regression Modeling Strategies: With Applications to Linear Models, Logistic and Ordinal Regression, and Survival Analysis*.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. Springer New York.

Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied logistic regression* (Third edition). Wiley.

Huang, J., & Ling, C. X. (2005). Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, *17*(3), 299–310.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An Introduction to Statistical Learning: With Applications in R*. Springer US.

Kingma, D. P., & Ba, J. (2017). Adam: A Method for Stochastic Optimization. *ArXiv:1412.6980 [Cs]*.

Kittler, J. (1998). Combining classifiers: A theoretical framework. *Pattern Analysis and Applications*, *1*(1), 18–27.

Kuhn, M., & Johnson, K. (2013). *Applied predictive modeling*. Springer.

Mang, C., & Chen, Y. (2020). Research on Flight delay Prediction based on Multi-Model Fusion. *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC)*, 720–725.

Nti, I. K., Adekoya, A. F., & Weyori, B. A. (2020). A comprehensive evaluation of ensemble learning for stock-market prediction. *Journal of Big Data*, *7*(1), 20.

Opitz, D., & Maclin, R. (1999). Popular Ensemble Methods: An Empirical Study. *Journal of Artificial Intelligence Research*, *11*, 169–198.

Polikar, R. (2006). Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, *6*(3), 21–45.

Polikar, R. (2012). Ensemble learning. In *Ensemble Machine Learning: Methods and Applications* (pp. 1–34).

Sagi, O., & Rokach, L. (2018). Ensemble learning: A survey. *WIREs Data Mining and Knowledge Discovery*, *8*(4), e1249. https://doi.org/10.1002/widm.1249

Sternberg, A., Soares, J., Carvalho, D., & Ogasawara, E. (2021). A Review on Flight Delay Prediction. *Transport Reviews*, *41*(4), 499–528.

Tibshirani, R. (1996). Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267–288.

Tibshirani, R. (2011). Regression shrinkage and selection via the lasso: A retrospective: Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *73*(3), 273–282.

Ting, K. M., & Witten, I. H. (1999). Issues in Stacked Generalization. *Journal of Artificial Intelligence Research*, *10*, 271–289.

van Loon, W., Fokkema, M., Szabo, B., & de Rooij, M. (2020). View selection in multi-view stacking: Choosing the meta-learner. *ArXiv:2010.16271 [Cs, Stat]*.

Vilalta, R., & Drissi, Y. (2002). A Perspective View and Survey of Meta-Learning. *Artificial Intelligence Review*, *18*(2), 77–95.

Wang, F., Bi, J., Xie, D., & Zhao, X. (2022). Flight delay forecasting and analysis of direct and indirect factors. *IET Intelligent Transport Systems*, *16*(7), 890–907.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, *5*(2), 241–259.

Zhang, C., & Ma, Y. (Eds.). (2012). *Ensemble Machine Learning*. Springer US.

Zhang, H., Yi, J., Liu, H., Zhong, G., & Li, G. (2021). Flight Delay Classification Prediction Based on Stacking Algorithm. *Journal of Advanced Transportation*, *2021*, 1–10.

Zhou, Z.-H. (2012). *Ensemble Methods: Foundations and Algorithms*. Chapman and Hall/CRC.

Appendix A - Variable Description

Name	Type	Description
arr_delay	categorical	Response, flight delayed or on_time
cos_day	numerical	Cosine transformed day (all dates are scheduled arrival times)
cos_hour	numerical	Cosine transformed hour
cos_month	numerical	Cosine transformed month
cos_week	numerical	Cosine transformed week
cos_year	numerical	Cosine transformed year
dest	categorical	Arrival airport
dest_awnd	numerical	Average daily wind speed (tenths of meters per second) at destination
dest_flght_day	numerical	Number of flights at destination
dest_prcp	numerical	Precipitation (tenths of mm) at destination
dest_snow	numerical	Snowfall (mm) at destination
dest_snwd	numerical	Snow depth (mm) at destination
dest_tmax	numerical	Maximum temperature (tenths of degrees C) at destination
dest_tmin	numerical	Minimum temperature (tenths of degrees C) at destination
dest_wsf2	numerical	Fastest 2-minute wind speed (tenths of meters per second) at destination
dest_wsf5	numerical	Fastest 5-second wind speed (tenths of meters per second) at destination
dest_wt01	categorical	Fog, ice fog, or freezing fog (may include heavy fog) at destination
dest_wt02	categorical	Heavy fog or heaving freezing fog (not always distinguished from fog) at destination
dest_wt03	categorical	Thunder at destination
dest_wt04	categorical	Ice pellets, sleet, snow pellets, or small hail at destination
dest_wt05	categorical	Hail (may include small hail) at destination
dest_wt06	categorical	Glaze or rime at destination
dest_wt07	categorical	Dust, volcanic ash, blowing dust, blowing sand, or blowing obstruction at destination
dest_wt08	categorical	Smoke or haze at destination
dest_wt09	categorical	Blowing or drifting snow at destination
dest_wt10	categorical	Tornado, waterspout, or funnel cloud at destination
dest_wt11	categorical	High or damaging winds at destination

Name	Type	Description
dest_wt18	categorical	Snow, snow pellets, snow grains, or ice crystals at destination
distance	numerical	"Distance between airports (miles)
holiday_dist	numerical	Time distance to the closest holiday in days
origin	categorical	Departure airport
org_awnd	numerical	Average daily wind speed (tenths of meters per second) at origin
org_flght_day	numerical	Number of flights at origin
org_prcp	numerical	Precipitation (tenths of mm) at origin
org_snow	numerical	Snowfall (mm) at origin
org_snwd	numerical	Snow depth (mm) at origin
org_tmax	numerical	Maximum temperature (tenths of degrees C) at origin
org_tmin	numerical	Minimum temperature (tenths of degrees C) at origin
org_wsf2	numerical	Fastest 2-minute wind speed (tenths of meters per second) at origin
org_wsf5	numerical	Fastest 5-second wind speed (tenths of meters per second) at origin
org_wt01	categorical	Fog, ice fog, or freezing fog (may include heavy fog) at origin
org_wt02	categorical	Heavy fog or heaving freezing fog (not always distinguished from fog) at origin
org_wt03	categorical	Thunder at origin
org_wt04	categorical	Ice pellets, sleet, snow pellets, or small hail at origin
org_wt05	categorical	Hail (may include small hail) at origin
org_wt06	categorical	Glaze or rime at origin
org_wt07	categorical	Dust, volcanic ash, blowing dust, blowing sand, or blowing obstruction at origin
org_wt08	categorical	Smoke or haze at origin
org_wt09	categorical	Blowing or drifting snow at origin
org_wt10	categorical	Tornado, waterspout, or funnel cloud at origin
org_wt11	categorical	High or damaging winds at origin
org_wt18	categorical	Snow, snow pellets, snow grains, or ice crystals at origin
p_airline	categorical	Airline
p_flights	numerical	Number of flights the plane had that date
p_manufac	categorical	Manufacturer of the plane
sin_day	numerical	Sine transformed day
sin_hour	numerical	Sine transformed hour
sin_month	numerical	Sine transformed month

Name	Type	Description
sin_week	numerical	Sine transformed week
sin_year	numerical	Sine transformed year