Mini Project 1

Machine learning (CS582), MIU [1]

Baraa Mousa Noufal   Ivan Krasowski Bissio

July 25, 2021

[1]Instructor: Anthony Sander

Contents

Abstract

1 About the Dataset

The chosen dataset collects information about Airline Passenger Satisfaction. We opted for this dataset because it had a lot of entries (almost 130000), many columns (including 4 categorical) and it was oriented to a binary classification problem ('satisfied' vs. 'neutral or dissatisfied').

1.1 EDA: Exploratory Data Analysis

- The dataset collects multiple features related to each passenger-flight, including labels for whether the passenger was satisfied with the flight or not.
- It has 24 columns (22 + id + result).
- Id column is dropped (it would add weight).
- Observation: the only column with null values is 'arrival_delay_in_minutes', with 393 missing values.
- Categorical columns: 4

  Features and possible values: Gender (2), customer_type (2), type_of_travel (2), customer_class (3)

  Decision: use one-hot encoding in all of them (none of them is a clear candidate for being weighted)
- Looking for strong correlations: pairwise correlation function to check if two features show strong correlation.

  NOTE: 'satisfaction' (the label we will try to predict) is not strongly correlated (>0.7) with any of the other features.

  arrival_delay_in_minutes and *departure_delay_in_minutes* have the highest correlation rate (0.965291), which semantically makes sense; all other variables are less than 75% correlated.

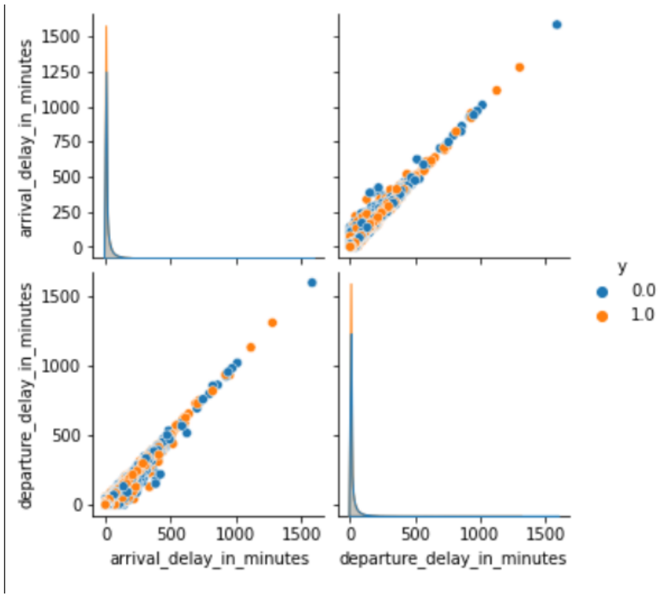

Figure 1: Correlation between most-correlated features

1.2 Data Preparation

1. X: all table except id and result ('satisfaction') columns; y: 'satisfaction' column
2. Since the arrival_delay feature is highly correlated with the departure_delay feature, and the missing values are not that many (393 out of 129879: 0.03%), we decide to remove the column.
3. We see there are 4 categorical features, with no more than 3 unique values each. So, given that none of them is a clear candidate for being weighted, we decide to use one-hot encoding in all of them.
4. Finally, we split X and y for training and validating, following a ratio of 80%/20% (the dataset is large enough).

   Variables: ***X_train, X_val, y_train, y_val***

2 Models

2.1 KNN

2.1.1 Presentation

KNearestNeighbors algorithm works by holding instances of training data and classifying by issuing a majority vote across "k" nearest neighbor of each point as to which class it belongs.

2.1.2 Defining Parameters

The data for training and validating is already defined by the Data Preparation step (80% train, 20% validation).
The representative parameter for KNN is *n_neighbors*: The number of neighboring instances taking part in the classification vote.

We trained the model on the default parameters (n_neighbors=5) and achieved a 77% accuracy on validation data.
We then attempted to tune the *n_neighbors* parameter using GridSearch across a range of values, it turned out the best value found by GridSearch for *n_neighbors* was 5 as well.
Next, we plotted the model score over a similar range, but scoring on the test dataset (rather than GridSearch).

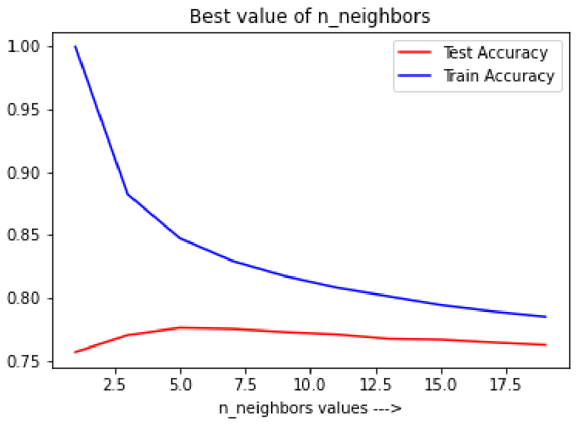

Figure 2: Complexity Curve: *n_neighbors* value for KNN

Observing the plot, it shows the same result as the validation accuracy starts to dip after *n_neighbors*=5.

2.1.3 Model Evaluation

Setting *n_neighbors*=5, we train another KNNClassifier and plot its learning curve as the data increases.

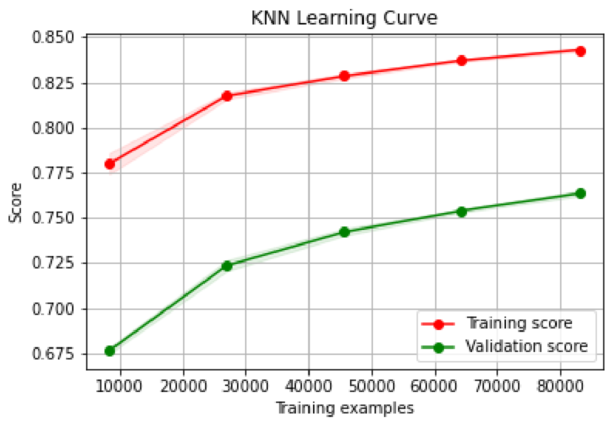

Figure 3: Learning Curve for KNN

Plot shows high variance but decreasing bias with more training data.

2.2 Decision Tree

2.2.1 Presentation

Decision Trees are a supervised learning model.
Decision Trees work by constructing a tree of features and value ranges, then traversing this tree to reach the classes at the leaf, they try to keep this tree balanced, but it isn't always.
The main hyperparameter to tune is the depth of this tree.

2.2.2 Defining Parameters

The data for training and validating is already defined by the Data Preparation step (80% train, 20% validation).
Representative parameter for this model is *max_depth*: The maximum depth of the constructed decision tree, the deeper the tree, the more likely it is to overfit the data.
Applying the algorithm with default *max_depth* (None) resulted in 94% accuracy.
Tuning *max_depth* with GridSearch algorithm over a range of values resulted in highest accuracy achieved at *max_depth*=13.
Applying the algorithm again over that same range, but validating using the test dataset shows very similar results, as shown in the plot below.

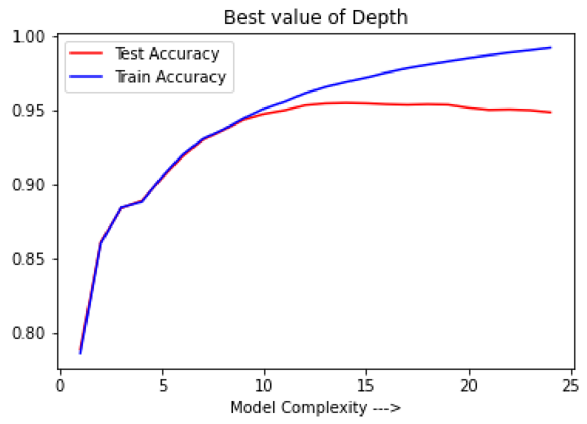

Figure 4: Complexity Curve: *max_depth* value for DecisionTreeClassifier

The model starts to overfit the training data after *max_depth*=13

2.2.3 Model Evaluation

Using the parameter *max_depth*=13, we plot the learning curve showing test and validation scores

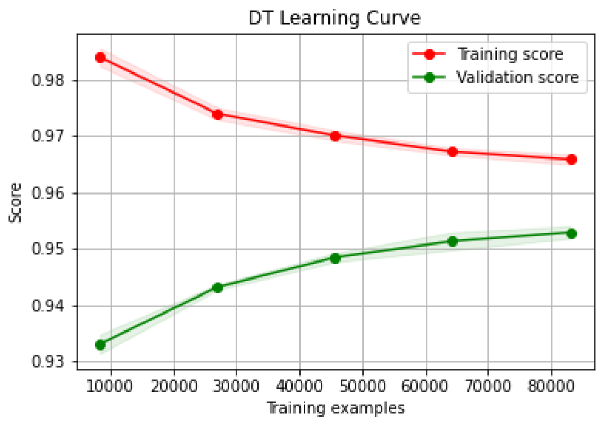

Figure 5: Learning Curve for DT

Both the variance and bias are lowering as we use more training data.
Observing the learning curve we see that the results start to converge, increasing the size of the dataset may increase the accuracy.

2.3 Support Machine Vector

2.3.1 Presentation

Support Vector Machines are a model of supervised learning.
In summary, the model finds the hyperplane (kernel) that maximizes the Margin of Safety for classifying.

2.3.2 Defining Parameters

The data for training and validating is already defined by the Data Preparation step (80% train, 20% validation).
Representative parameters for the model are *gamma* and C

- gamma: kernel coefficient
- C: regularization parameter (higher C, higher variance)

We train the following values for the *gamma* parameter: [0.001, 0.01, 0.03, 0.05, 0.08, 0.1, 0.5]
To improve the predictions, we regularize the data using a StandardScaler (removes the mean and scales to unit variance) The training error and the validation error for each value allow us to plot a Complexity Curve, to select the optimal value fo the parameter.

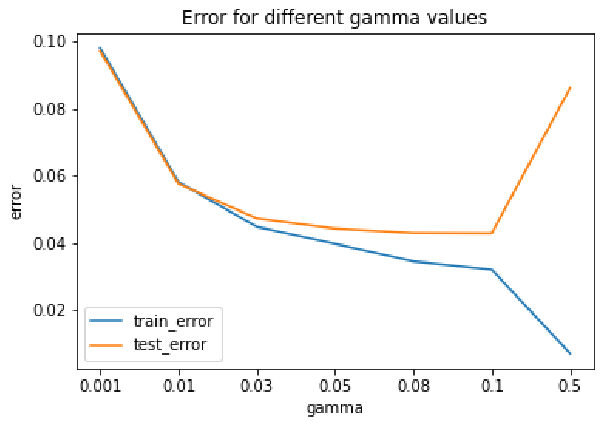

Figure 6: Complexity Curve: *gamma* value for SVM

By observing the plot, we determine that the best value for gamma is 0.03
We train the following values for the C parameter: [0.02, 0.2, 0.8, 1.2, 2, 5, 10]
Again, we regularize the data using a StandardScaler (removes the mean and scales to unit variance) The training error and the validation error for each value allow us to plot a new Complexity Curve, to select the optimal value fo the parameter.

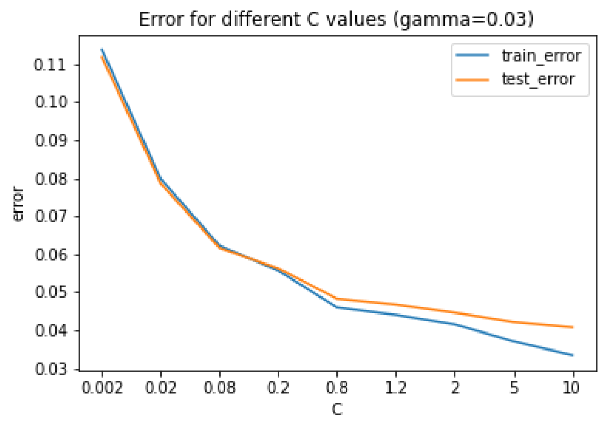

Figure 7: Complexity Curve: C value for SVM ($gamma$=0.03)

By observing the plot, we determine that the best value for C is 0.8

2.3.3 Model Evaluation

Chosen the parameters: ($gamma$=0.03, C=0.8), a learning curve shows us the training and validation scores for different data sizes.
This way, we are able to say that the score is bounded below 95%, and the model doesn't seem to continue learning after 65000/70000 training rows.

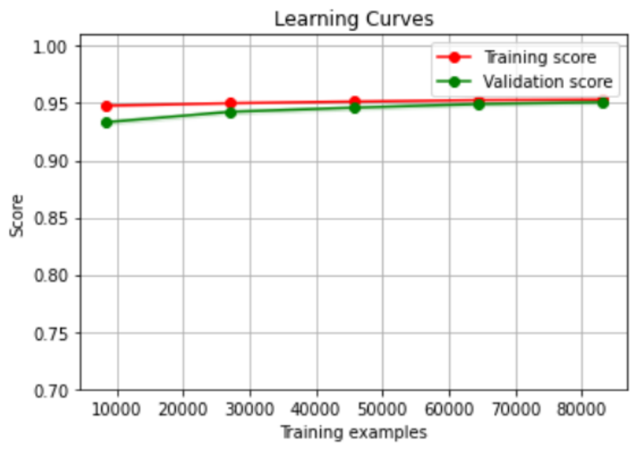

Figure 8: Learning Curve for SVM

2.4 Neural Network

2.4.1 Presentation

Neural Networks (or Multilayer Perceptrons) are a model of supervised learning, represented by composed non-linear functions.

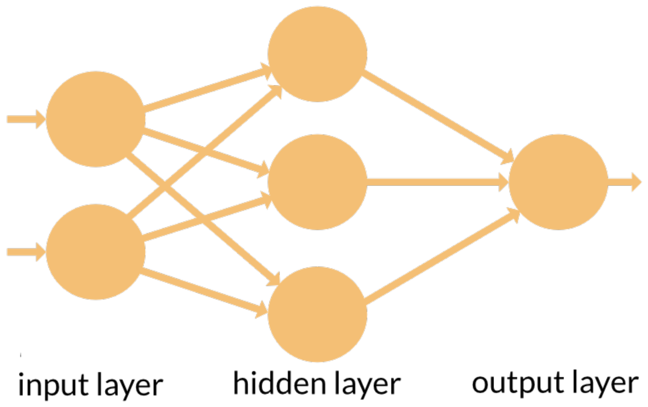

Figure 9: Multilayer Perceptron / Neural Network [1]

They consist of ”neurons”, each of which represents a non-linearity, divided in layers, each of which depends on the previous (the first one, on the input data) and is crucial for the next one (the last one gives the output)
They can be seen as a function, whose interface is defined by the ”input layer” (the first layer) and the ”output layer” (the result).
All layers between the input layer and the output layer are ”hidden layers”.
A Neural Network makes its predictions after having its parameters (weights) trained with labeled example data (train data)

2.4.2 Defining Hyperparameters

The data for training and validating is already defined by the Data Preparation step (80% train, 20% validation).
Representative hyperparameters (different from the parameters to be trained) for the model are *alpha* and *learning_rate*

- alpha: regularization parameter (L2 penalty).
- learning_rate: parameter for controlling the step size when updating the weights.

A Grid Search is used for getting the best combination of parameters (alpha, learning_rate) for the input and the model.
Values used for *alpha*: [0.001, 0.005, 0.01, 0.05, 0.1]
Values used for *learning_rate*: [0.001, 0.005, 0.01, 0.05, 0.1]

[1]https://appliedgo.net/perceptron/

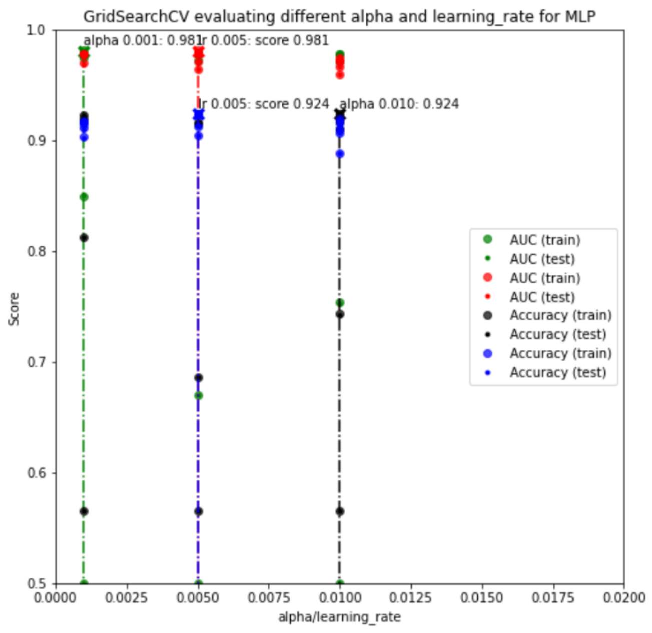

Figure 10: Grid Search results

Observable on the plot, and also obtainable by requesting the *grid_search.best_params_*, the best values for the hyperparameters are:

- alpha: 0.001
- learning_rate: 0.005

2.4.3 Model Evaluation

Chosen the parameters: (*alpha*=0.001, *learning_rate*=0.005), a learning curve shows us the training and validation scores for different data sizes.
This way, we are able to say that the model seems to continue learning after 90000 training rows.

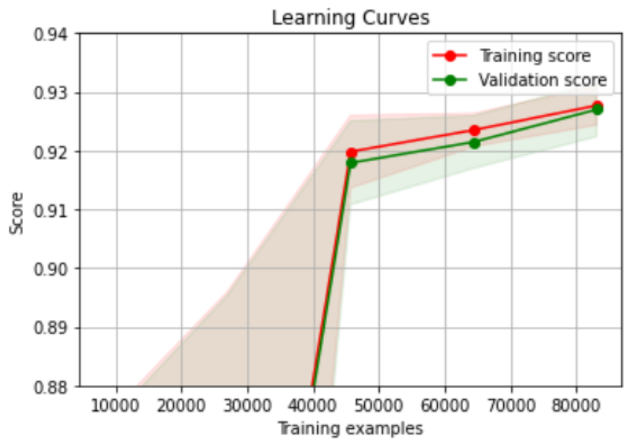

Figure 11: Learning Curve for MLP

2.5 Stochastic Gradient Descent

2.5.1 Presentation

Stochastic Gradient Descent Classifier is an optimization technique to train other classification models, depending on chosen loss function it can be equivalent to an SVM or Logistic Regression models, it is particularly useful for large and sparse data.

2.5.2 Defining Parameters

The data for training and validating is already defined by the Data Preparation step (80% train, 20% validation).
Representative parameters for the model are *max_iter*, *loss* and *alpha*.

- max_iter: stopping criterion, the algorithm can stop before, but never after this number of iterations.
- loss: loss function, "hinge" is equivalent to a linear SVM, "log" is logistic regression.
- alpha: regularization parameter (higher alpha, higher variance)

We trained *alpha* over a logarithmically decreasing range [1, 0.1, 0.001...10**-7], *max_iter* over the range [10000, 100000] increasing 10000 per step, as well as across both "hinge" and "log" loss functions.
GridSearch was used with those ranges to determine best values for the hyperparameters, it was found to be alpha=0.01, loss=hinge, and max_iter=60000.
To improve the predictions, we regularize the data using a StandardScaler. The training error and the validation error for each value allow us to plot a Complexity Curve, to select the optimal value fo the parameter. Setting *max_iter*=60000, *loss*="hinge", we plot the validation error across a range of values of *alpha*

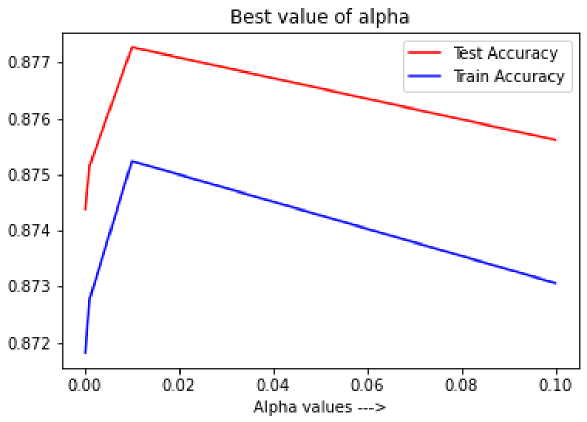

Figure 12: Complexity Curve: *alpha* value for SGD

2.5.3 Model Evaluation

With *alpha*=0.01 we plotted a learning curve for the model as the sample size grows.

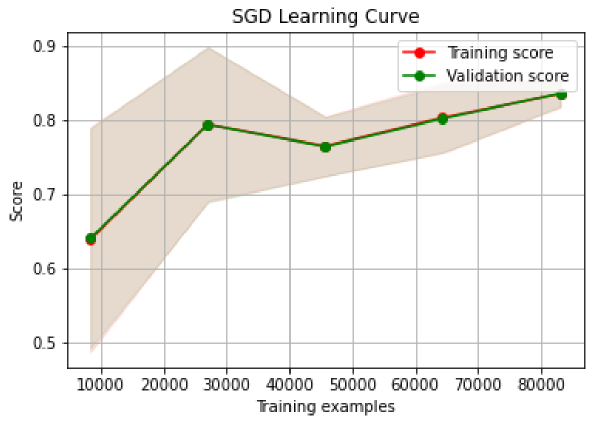

Figure 13: Learning Curve for SGD

This model shows extremely low bias and variance, and an increasing validation score as the data increases.
It is also important to note that this model only took a few seconds to train, which makes it much faster to train than the equivalent linear SVM, and much better suited for very large data.

2.6 Ensemble: Random Forest

2.6.1 Presentation

Ensemble learning models combine results of multiple models to improve overall performance.
There are many different techniques in which to apply this concept, we will be using a RandomForestClassifier, which falls under the bagging ensemble techniques.
Bagging is simply the idea of splitting the data into subsets and training a model on each subset, then combining the results to achieve a more generalized result, a RandomForestClassifier uses Decision Trees as the internal models.

2.6.2 Defining Parameters

The data for training and validating is already defined by the Data Preparation step (80% train, 20% validation).
Representative parameters for the model are *n_estimators*, *max_features*, and *max_depth*.

- n_estimators: The number of internal models used in the ensemble.
- max_features: The maximum number of features allowed for the split in each decision tree.
- max_depth: The maximum depth of each tree in the ensemble.

We trained an ensemble using a GridSearch over a few different values of these parameters to see if we can achieve a higher accuracy than the defaults.
We achieved 96.4% accuracy with the parameters set to 'bootstrap': True, 'max_depth': 50, 'max_features': 10, 'n_estimators': 200, only a slightly (0.2%) higher accuracy score than default parameters. Tuning this model proved difficult as it was overfitting the data very fast, as we can see from this complexity curves over different values of *n_estimators* and *max_depth*.

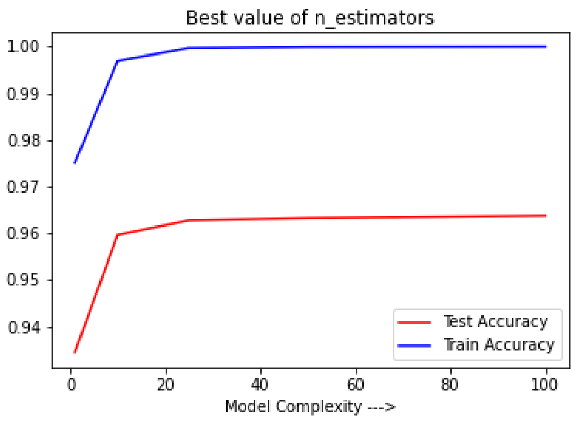

Figure 14: Complexity Curve: *n_estimators* value for RFC

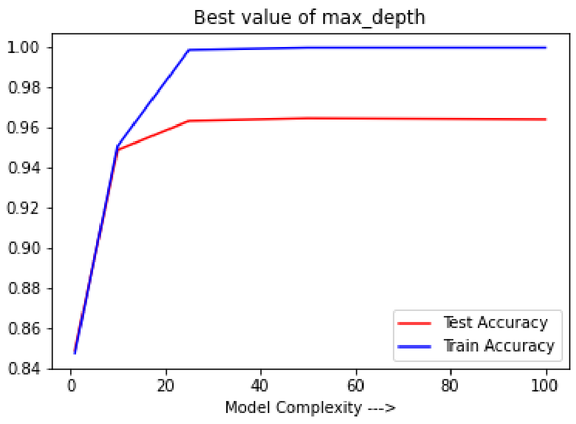

Figure 15: Complexity Curve: *max_depth* value for RFC

2.6.3 Model Evaluation

We plotted the learning curve for the model using default params, over the training data split (around 90k rows).

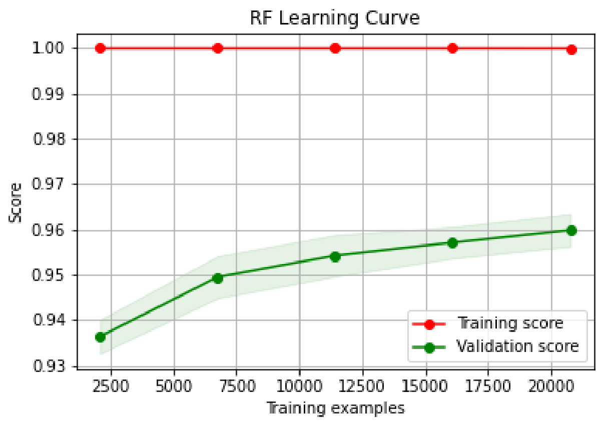

Figure 16: Learning Curve for RFC over large dataset

It shows a very high bias and an almost immediate overfitting, so as an experiment we plotted the learning curve over a smaller data split (in this case our validation split, about 30k rows) it showed how quickly the model overfits the training data.

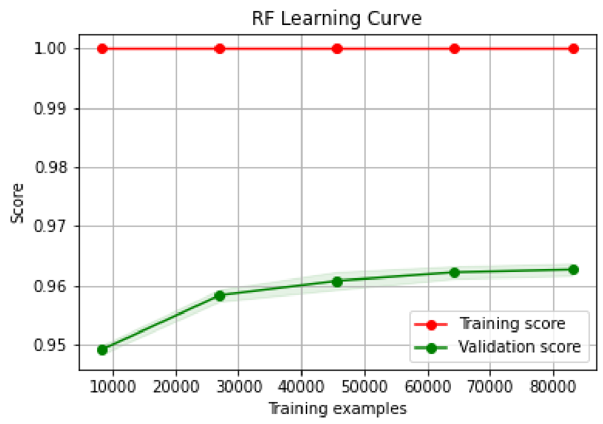

Figure 17: Learning Curve for RFC over small dataset

2.7 Voting Classifier

3 AUCs

3.1 ROC curve

The ROC (Receiver Operating Characteristics) curve is a plot which shows the performance of a binary classifier as its discrimination threshold is varied.
The curve is created by plotting the true positive rate vs the false positive rate at various threshold settings.

3.2 ROC plot of trained models

Below is the ROC plot of all the trained models.

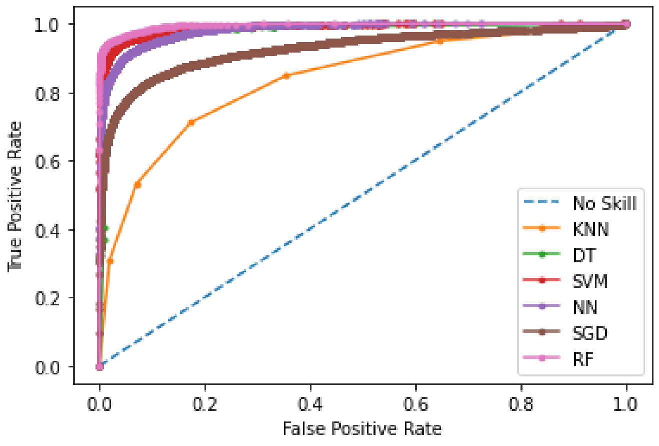

Figure 18: ROC Curve of all trained models

The AUC (Area Under Curve) for each model was calculated to be

- KNN=0.837
- DT=0.981
- SVM=0.988
- NN=0.982
- SGD=0.927
- RF=0.995

4 AutoML

4.1 Presentation

AutoML is a tool for choosing the best model (or ensemble) for fitting and predicting a dataset.
If using scikit-learn, the proper toolkit is defined as auto-sklearn.

4.2 Obtaining the Model

Simply fitting the training data to the AutoSklearnClassifier gives us an "optimal" model.
In this case (shown by *running model.show_models*()), the result is an ensemble of several ensembles, involving SVMs, Random Forests, etc.

4.3 AUC vs best model

We calculated the AUC curve from plotting the AutoSKlearnClassifier vs our best model, the Voting Classifier,

5 Best model: model n

5.1 PCA

6 Conclusion