Titanic : Exploration and survival prediction

An De Rijdt[1]

[1] VU Amsterdam
a.j.de.rijdt@student.vu.nl

Abstract. In this assignment we explore the titanic dataset and predict survival rate based on passenger features like age, gender and fare. The analysis was performed using python in a jupyter notebook.

Keywords: Data mining, Titanic, Classification

1 Data exploration

1.1 Loading the data

We load the titanic dataset in a pandas dataframe and inspect the different features.

1.2 Exploring the data: the first steps

We examine some basic statistics for the numerical and categorical For the categorical variables we see that people shared a cabin and the most people embarked in Southhampton.

Table 1. Statistics for the non-numeric features

	Name	Sex	Ticket	Cabin	Embarked
count	891	891	891	204	889
unique	891	2	681	147	3
top	Chibnall, Mrs. (Edith Martha Bowerman)	male	1601	C23 C25 C27	S
freq	1	577	7	4	644

We replace the zero Fares with NaN to be able to impute them later. We make a new feature family size and drop name and ticket columns. We split the variables in numberic and categorical to treat them differently in the plots later.

We make histograms and boxplots of the different numerical columns. We see that Fare is very skewed, which is normal, many people bought a cheap ticket and very few bought an expensive one.

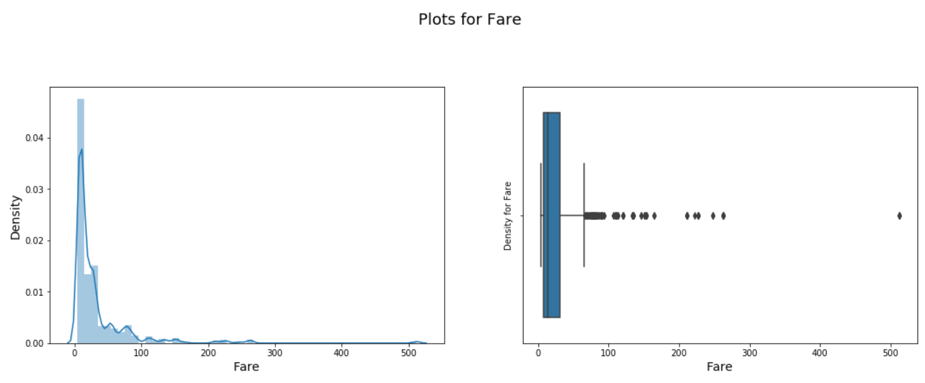


Fig. 1. The Fares are very skewed. We will have to scale them later before we fit the classifiers.

We make a simple countplot for each variable to see how the passengers are distributed over the different categories. We see for instance that more males than females where on board and many people travelled alone.

1.3 Treating missing values

We see that there are only missing values in Age, Cabin and Embarked. Age is not too skewed so we impute by the mean. We drop the Cabin column as there are soo many missing values.

We see that all the missing Fares are Embarked at Southampton. It seems reasonable that fare is dependent on where you embark and the class. As the fares are skewed, we use imputation by the median per class.

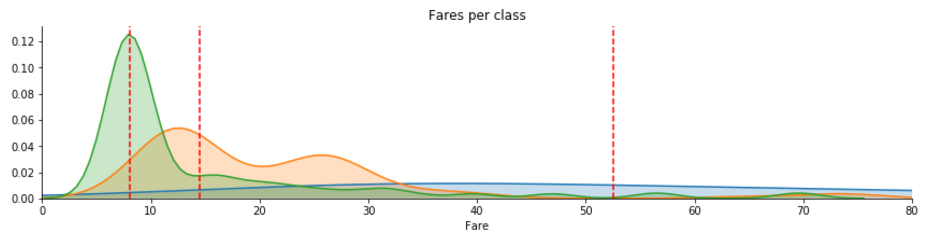


Fig. 2. Fares per class for embarkment in Southampton.

In the same way we derive that the 2 passengers missing embarkment have most likely embarked in Charlottesville.

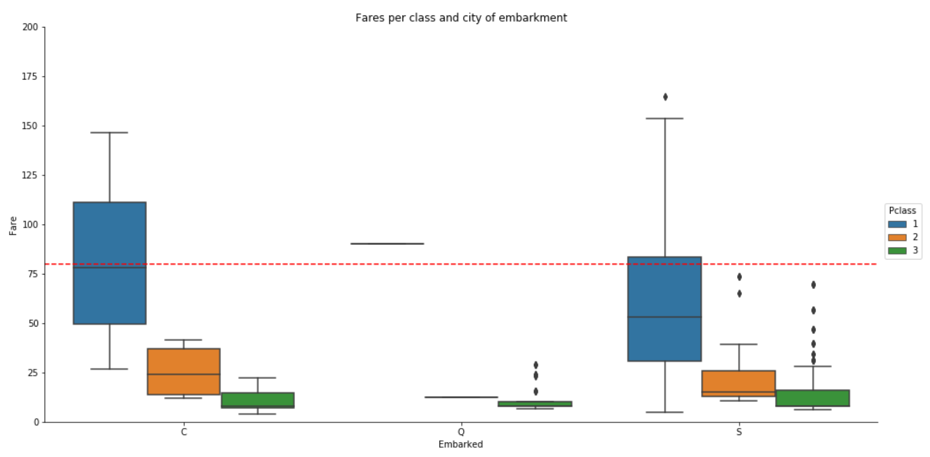


Fig. 3. We derive that the passengers have probably embarked in Charlottesville..

1.4 Correlations

We now explore the correlation between the by generating a correlation matrix. We see that class and fare are negatively correlated, as we also noticed when we were imputing the null values.

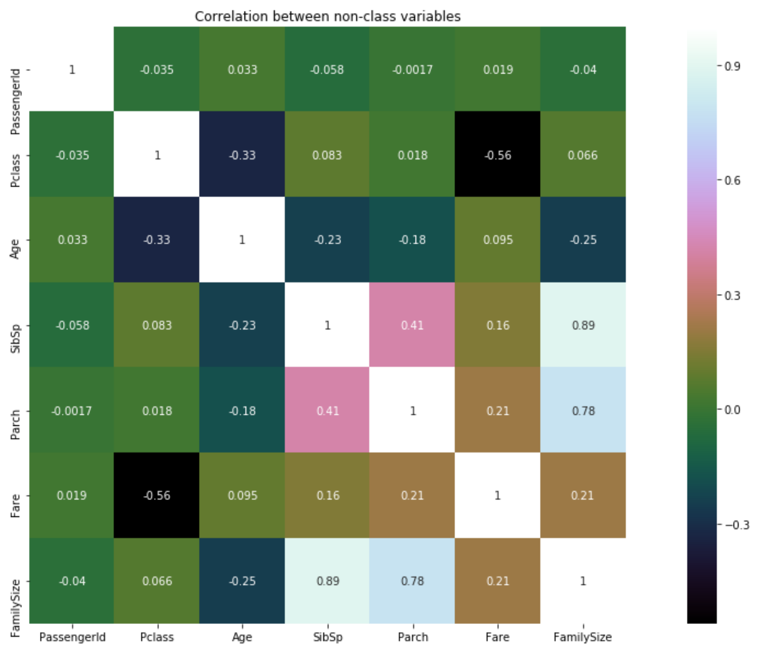


Fig. 4. Class and fare are highly correlated.

The highest correlation with the class variable can be seen in the passenger class and fare. Those are also very correlated amongst each other.

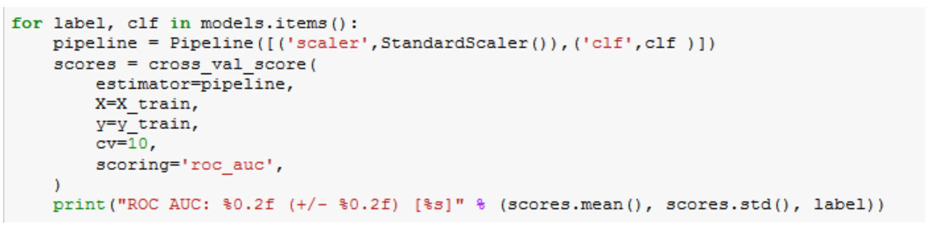
2 Classification

2.1 Different classifiers

We split our dataset in train and testset and apply and compare different classifiers. As mentioned before, we have to center and scale age and fare. We hence setup an scikit-learn pipeline with the scaler and the model to try. We make a dictionary of models to try:

- logistic regression
- support vector machines
- Random Forest
- K nearest neighbours for different numbers of neighbours.

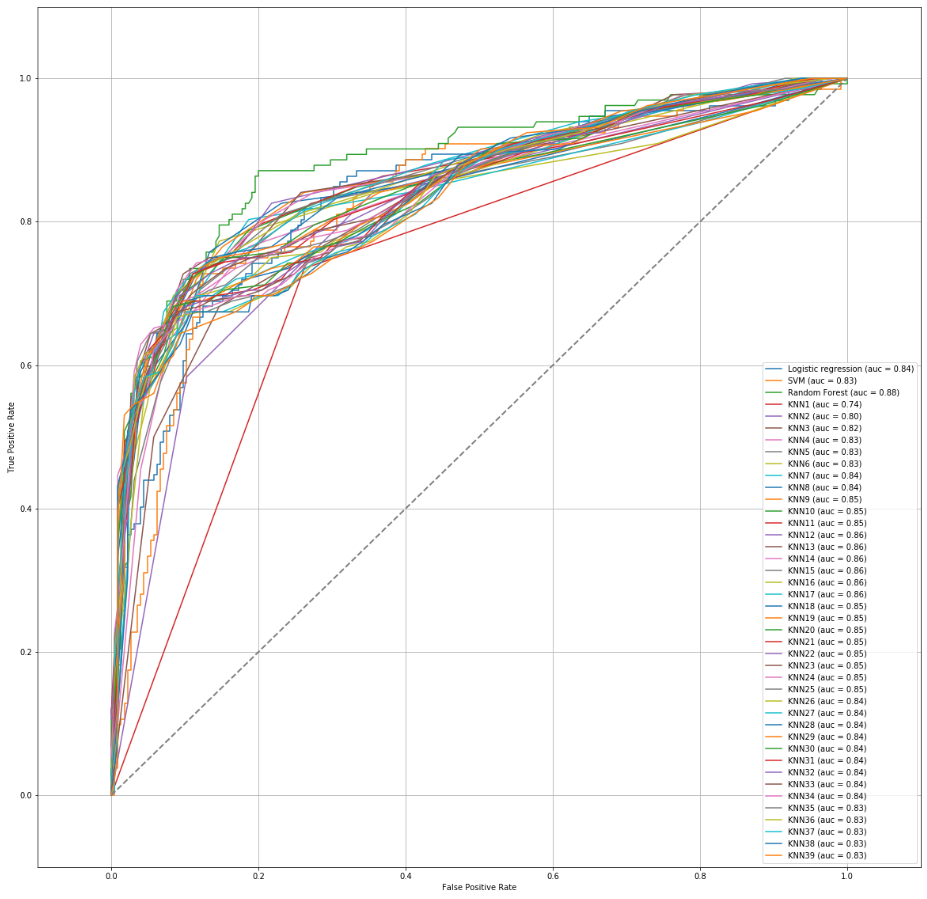
We use cross validation on the trainset.



```
for label, clf in models.items():
    pipeline = Pipeline([('scaler',StandardScaler()),('clf',clf )])
    scores = cross_val_score(
        estimator=pipeline,
        X=X_train,
        y=y_train,
        cv=10,
        scoring='roc_auc',
    )
    print("ROC AUC: %0.2f (+/- %0.2f) [%s]" % (scores.mean(), scores.std(), label))
```

Fig. 5. Training the model.

We display area under the curve to compare the models on the testset. KNN works best with 23 neighbours works best but Random Forest beats all other models both on train and testset. Logistic regression has poor performance on the testset which is not surprising as we are dealing with highly correlated variables.



References

1. An De Rijdt: https://github.com/klimantje/DataMining