Data Mining in Smart Systems

Sara Fattouh [G9UHTN]

[1] Eötvös Loránd University, Hungary
2 Pázmány Péter stny. 1/C, Budapest, Hungary
www.elte.hu

Abstract. In this paper we describe our approaches on the given data, study the data and the task at hand. and propose various models and techniques. We will work on scores predicting.

Keywords: Dataset, Model, Random Forest, XGBoost, Decision Tree, Support Vector Classifier (SVC), K-nearest neighbor (KNN).

1 Introduction

In this paper we describe our analysis on the given dataset, study the data and the task, and propose various models and techniques. We will predict scores of hotels.

We will use the CRISP-DM methodology to describe our approaches used in model generating and dataset training.

2 Cross Industry Standard Process for Data Mining (CRISP-DM)

2.1 Business Understanding

Tripadvisor is an American online travel company that operates a website and mobile app with user-generated content, a comparison shopping website, and offers online hotel reservations as well as bookings for transportation, lodging, travel experiences, and restaurants.
In this paper we describe several approaches to predict hotels' scores using classification models.

2.2 Data Understanding

The given dataset includes quantitative and categorical features from online reviews of 21 hotels located in Las Vegas Strip, extracted from TripAdvisor. All the 504 reviews were collected between January and August of 2015.

The dataset contains 504 records and 20 tuned features, 24 per hotel (two per each month, randomly selected), regarding the year of 2015. Categorical attributes describe the traveler and review specifications (user country, user continent, period of stay, traveler type, review month, review weekday). Other Categorical attributes describe the hotel and its facilities (hotel name, hotel stars, Pool, Gym, Tennis court, Spa, Casino, Free internet).
There are numerical attributes in the dataset (Nr. Reviews, Nr. hotel reviews, Helpful votes, Nr. Rooms, Member years). The label attribute is the "Score" of the hotel for which we will use learning techniques to predict.

A question relating score is about hotel star. Does a hotel with a higher star hotel impresses the user more than hotel with lower star?
To have more insight into the data we made use of the seaborn visualization library to plot the relationship between Hotel stars and hotel Score attributes.

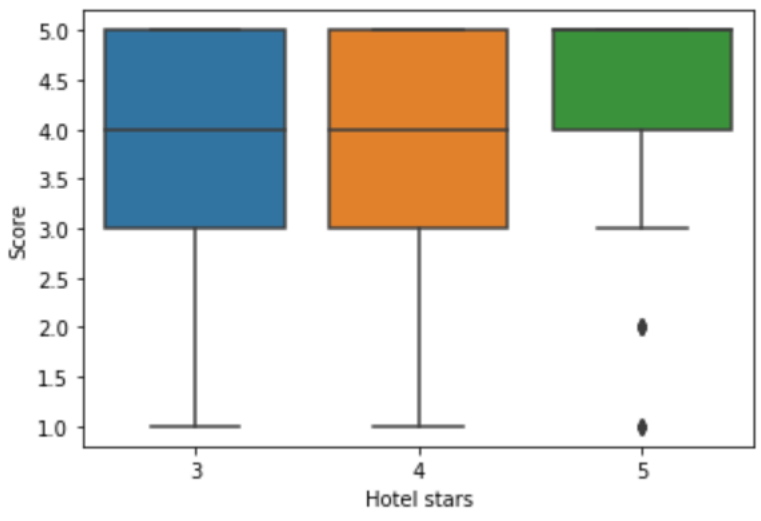


Figure 1 the relationship between Hotel stars and Score

Probably because all hotels are in Vegas, even hotels with lower hotel stars still provide good services to customers. Average score for each hotel star category is over 4. So there is no difference between hotels with higher stars and with lower stars

Another possible point of view is concerned with the facilities. In the dataset, there are six variables about facilities, Pool, Gym, Tennis court, Spa, Casino and Free internet. Let's have a look on whether the presence these facilities can boost users' satisfaction.

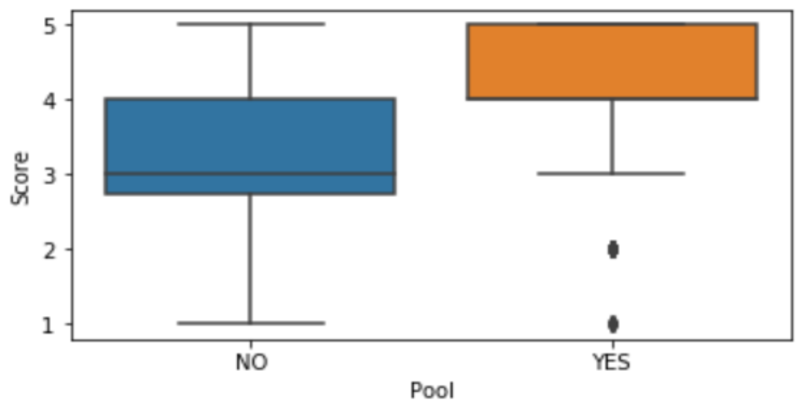


Figure 2 the relationship between Pool and Score

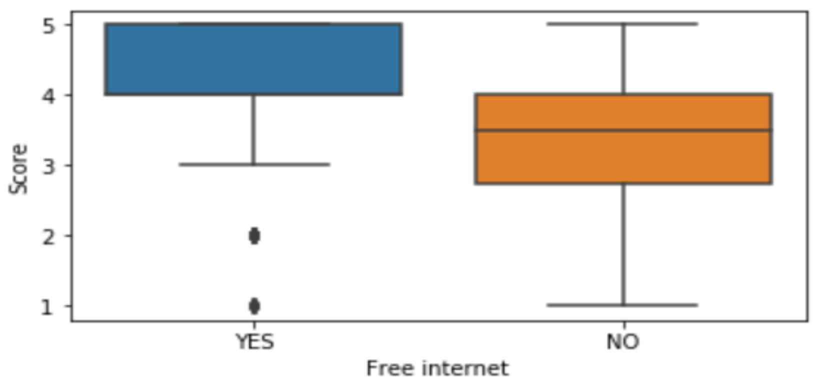


Figure 3 the relationship between Free internet and Score

We plotted the relationship between score and all other facilities in the same manner. Having looked at the graphs, we noticed that here is a decrease on score for hotel without pool and free internet (why would people stay in hotels without free Wi-Fi?).

As for other facilities, no significant impact on the score was found.

2.3 Data Preparation

Real world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends and is likely to contain many errors.
Considering the fact that high quality data leads to better models and predictions, data preprocessing became vital and the fundamental step in data science since it has a direct impact on the success rate of the entire project.

Checking for any Null or NA (Not Available) Values

It is necessary before any machine learning task to check for any missing data. Sometimes the description of the data might be enough to decide but it is better to validate it in the code. Pandas library provides two useful methods to check for any null or missing values, which are isnull(), and isna(). unfortunately, the dataset that we are using is not complete, and there are missing values in both training and testing datasets. we need to replace null values as a necessary step for cleaning the data before applying any classification algorithm. We choose to replace those null values by the most frequent value of each column.

Data Transformation

Transforming business data can ensure maximum data quality which is vital to gain precise analysis, leading to valuable insights that will ultimately reinforce data-driven decisions. Data transformation is undertaken with the intention to enhance the ability of the classification algorithms that we are going to apply on the dataset.

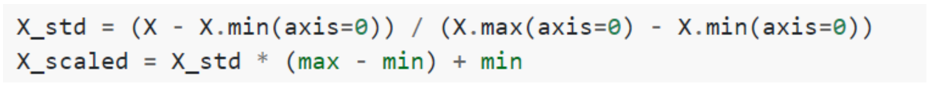
Convert Categorical Attributes to Numerical

Most of the machine learning algorithm cannot handle categorical data unless we convert them to numerical values. Las Vegas hotels data set contains 13 categorical attributes which need to be converted to numerical.
There are two options here. The first one is to use the Label Encoder which replaces the categorical attributes to numbers. However, the label encoding produces another problem which is, since there are different numbers in the same column, the model will misunderstand the data to be n some kind of order, $0<1<2$. But this is not the case at all. To overcome this problem, we use One Hot Encoding instead.

What one hot encoding does is, it takes a column which has categorical data, which has been label encoded, and then splits the column into multiple columns. The numbers are replaced by 1s and 0s, depending on which column has what value.

Features Scaling

Feature Standardization is an important preprocessing before applying any machine learning algorithm. The theory behind feature scaling is that the majority of classifiers use the Euclidean Distance to measure the similarity between Objects. So that if a feature exists with a broad range of values, then the distance measurement will be governed by this feature. Feature Standardization involves rescaling the features so that they have the properties of a Standard normal distribution (with mean equals 0 and a standard deviation of 1).
Since we have both numerical and categorical attributes, we are going to scale the numerical attributes. The scaler used is Min Max Scaler. It transforms features by scaling each feature to a given range. This estimator scales and translates each feature individually such that it is in the given range on the training set, e.g. between zero and one.

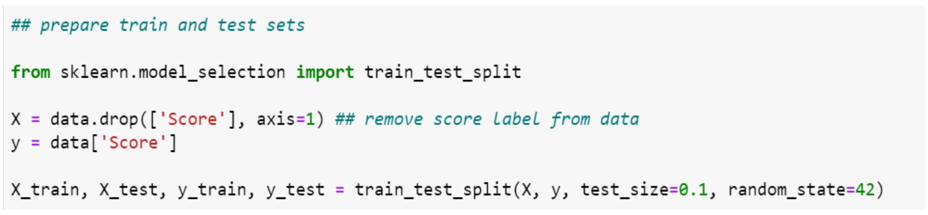


```
X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
X_scaled = X_std * (max - min) + min
```

Figure 4 MiniMax Scaler formula

Dataset Splitting

After data cleaning. It is time to split the dataset into random train and test subsets. The proportion of the dataset to include in the test split is 0.1 since our dataset is not very rich, we need to make use of more records in models training.



```
## prepare train and test sets

from sklearn.model_selection import train_test_split

X = data.drop(['Score'], axis=1) ## remove score label from data
y = data['Score']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)
```

Figure 5 Initializing the train and test datasets

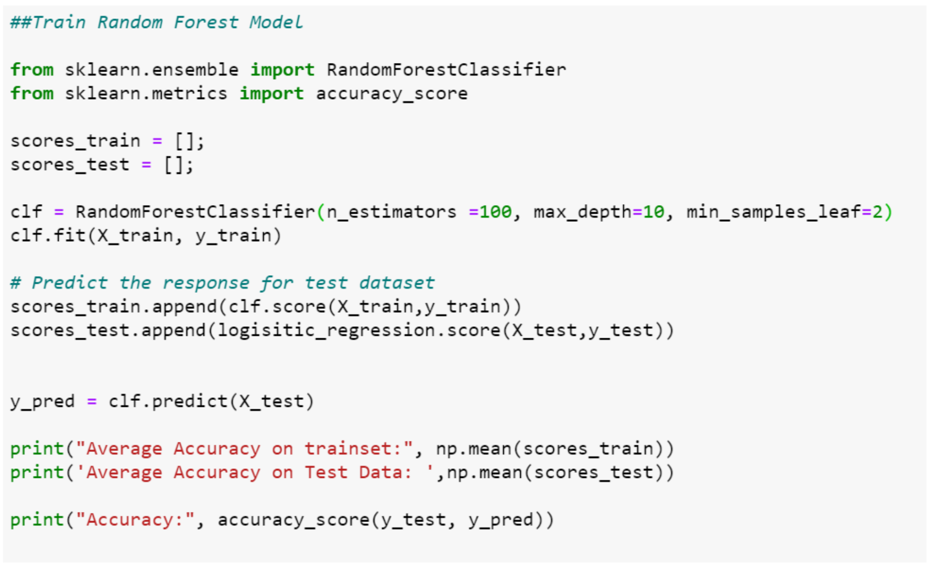
2.4 Modeling

Let us study the models that we have implemented on the data set and the different between them.

Random Forest

creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is said that the more trees it has, the more robust a forest is.
n_estimators: represents the number of trees in the forest. Usually the higher the number of trees the better to learn the data. However, adding a lot of trees can slow down the training process considerably.

- n_estimators: 100 estimators and this number of estimators gave reasonable accuracy.
- max_depth: represents the depth of each tree in the forest. The deeper the tree, the more splits it has and it captures more information about the data. We chose the value 10.



```
##Train Random Forest Model

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

scores_train = [];
scores_test = [];

clf = RandomForestClassifier(n_estimators =100, max_depth=10, min_samples_leaf=2)
clf.fit(X_train, y_train)

# Predict the response for test dataset
scores_train.append(clf.score(X_train,y_train))
scores_test.append(logisitic_regression.score(X_test,y_test))


y_pred = clf.predict(X_test)

print("Average Accuracy on trainset:", np.mean(scores_train))
print('Average Accuracy on Test Data: ',np.mean(scores_test))

print("Accuracy:", accuracy_score(y_test, y_pred))
```

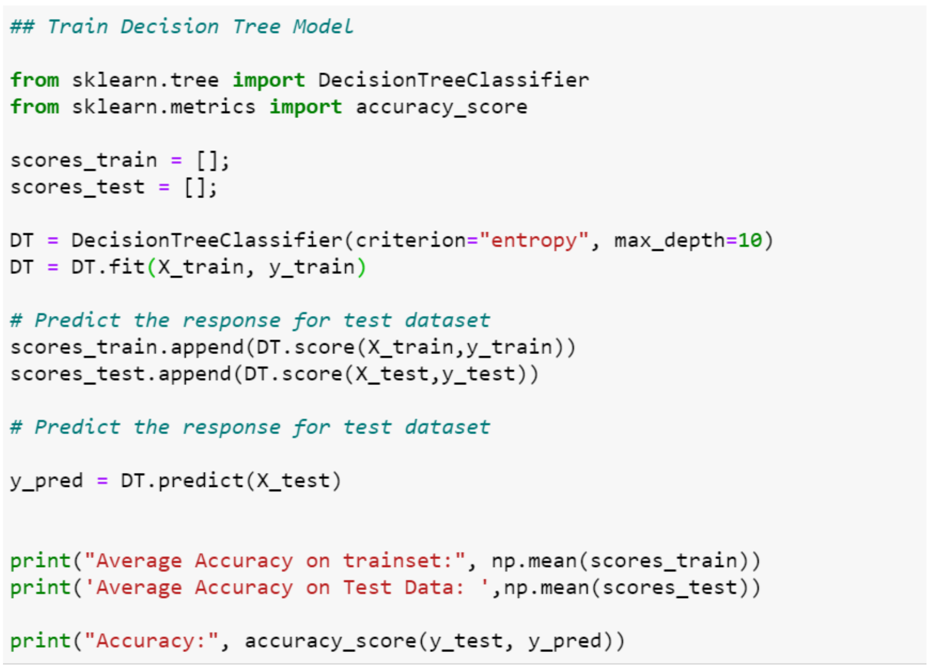
Figure 6 Random Forest Estimator

```
Average Accuracy on trainset: 0.739514348785872
Average Accuracy on Test Data:  0.45098039215686275
Accuracy: 0.45098039215686275
```

Decision Tree

The next model that we used is decision tree, which generates a model based on extracting decision rules form the training data set. It breaks down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

- criterion: This parameter determines how the impurity of a split will be measured. “entropy” is one of the methods used in decision tree algorithms to decide the optimal split from a root node, and subsequent splits which we used and showed better results than other criterions e.g. Gini index.
- Max_depth: represents the depth of the tree. 10 depth gave reasonable results.



```
## Train Decision Tree Model

from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

scores_train = [];
scores_test = [];

DT = DecisionTreeClassifier(criterion="entropy", max_depth=10)
DT = DT.fit(X_train, y_train)

# Predict the response for test dataset
scores_train.append(DT.score(X_train,y_train))
scores_test.append(DT.score(X_test,y_test))

# Predict the response for test dataset

y_pred = DT.predict(X_test)


print("Average Accuracy on trainset:", np.mean(scores_train))
print('Average Accuracy on Test Data: ',np.mean(scores_test))

print("Accuracy:", accuracy_score(y_test, y_pred))
```

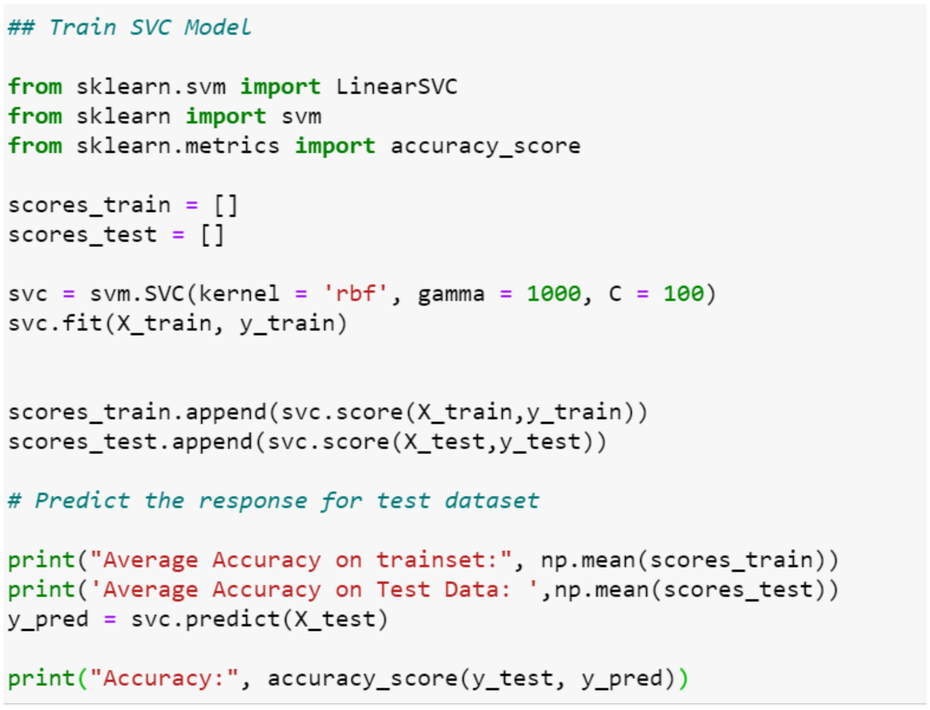
Figure 7 Decision Tree Estimator

```
Average Accuracy on trainset: 0.82560706401766
Average Accuracy on Test Data:  0.43137254901960786
Accuracy: 0.43137254901960786
```

Support Vector Classifier (SVC)

The objective of a Linear SVC (Support Vector Classifier) is to fit to the data we provide find the best hyperplane to separate the different classes by maximizing the distance between sample points and the hyperplane.

- Kernel parameter: selects the type of hyperplane used to separate the data. Using 'linear' will use a linear hyperplane (a line in the case of 2D data). 'rbf' and 'poly' uses a nonlinear hyper-plane. Since our data is not 2D we used the RBF, Radial Basis Function.
- Gamma: is a parameter for nonlinear hyperplanes. It defines how far the influence of a single training example reaches. The higher the gamma value it tries to exactly fit the training data set. We chose 1000 for this parameter.
- C: is the penalty parameter of the error term. It controls the tradeoff between smooth decision boundary and classifying the training points correctly. We used 100.
-



```python
## Train SVC Model

from sklearn.svm import LinearSVC
from sklearn import svm
from sklearn.metrics import accuracy_score

scores_train = []
scores_test = []

svc = svm.SVC(kernel = 'rbf', gamma = 1000, C = 100)
svc.fit(X_train, y_train)


scores_train.append(svc.score(X_train,y_train))
scores_test.append(svc.score(X_test,y_test))

# Predict the response for test dataset

print("Average Accuracy on trainset:", np.mean(scores_train))
print('Average Accuracy on Test Data: ',np.mean(scores_test))
y_pred = svc.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
```

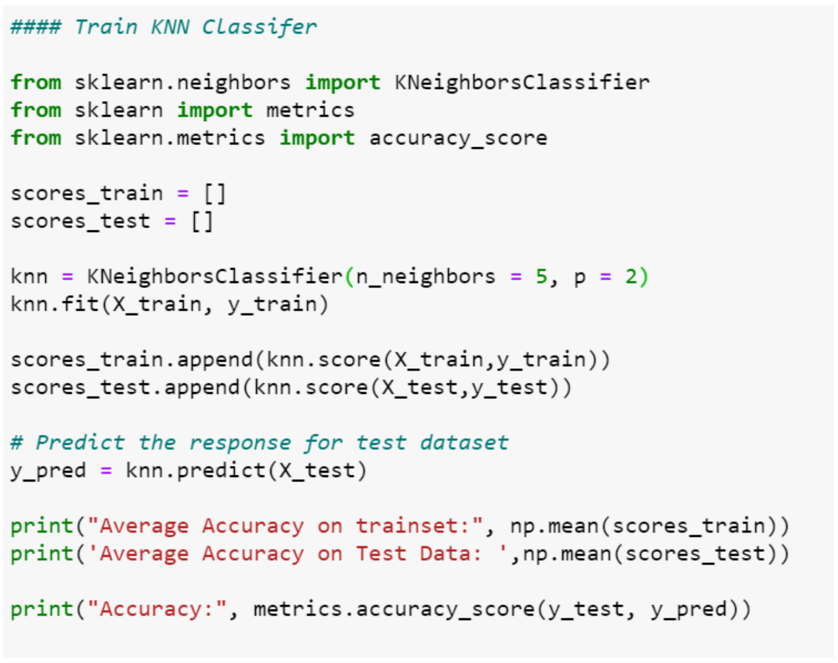
Figure 8 Support Vector Classifier

Average Accuracy on trainset: 0.9955849889624724
Average Accuracy on Test Data: 0.47058823529411
Accuracy: 0.47058823529411764

K-Nearest Neighbors (KNN)

KNN algorithm works to classify new data based on its proximity to K-neighbors (training data). So if the new data is surrounded by training data that has Class 1, it can be concluded that the new data is included in Class 1.

- n_neighbors: number of neighbors to consider. K = 5 was used in the model.
- P: distance metrics. P = 2 which represents the Euclidean distance gave better results than Manhattan and Minkowski.



```
#### Train KNN Classifer

from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.metrics import accuracy_score

scores_train = []
scores_test = []

knn = KNeighborsClassifier(n_neighbors = 5, p = 2)
knn.fit(X_train, y_train)

scores_train.append(knn.score(X_train,y_train))
scores_test.append(knn.score(X_test,y_test))

# Predict the response for test dataset
y_pred = knn.predict(X_test)

print("Average Accuracy on trainset:", np.mean(scores_train))
print('Average Accuracy on Test Data: ',np.mean(scores_test))

print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
```

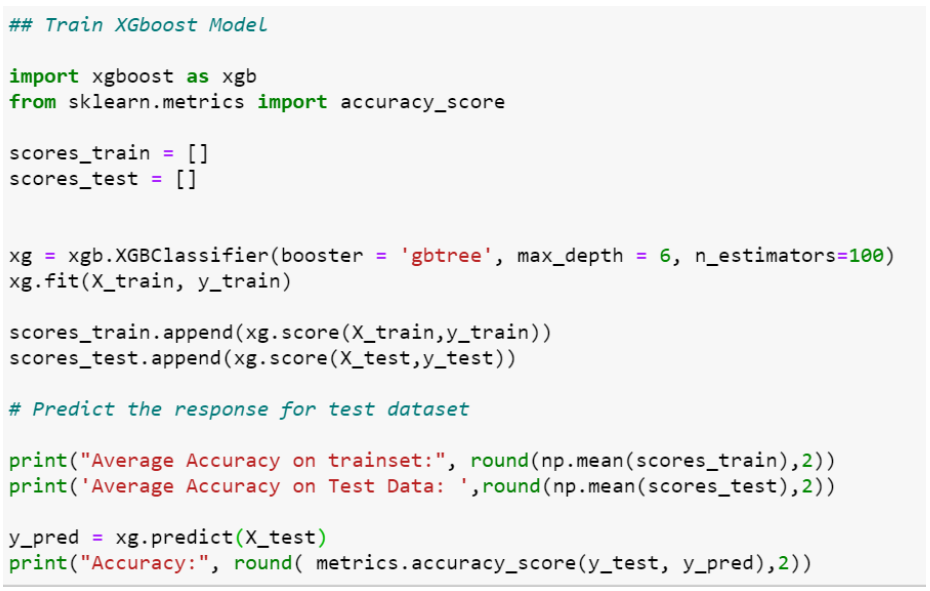
Figure 9 Classifying Data Using K Nearest Neighbors

Average Accuracy on trainset: 0.5938189845474614
Average Accuracy on Test Data: 0.45098039215686275
Accuracy: 0.45098039215686275

XGboost

Stands for Extreme Gradient Boosting; it is a decision tree algorithm that recently became dominant in Kaggle. It is an implementation of Gradient Boosting with improvements. Gradient boosting involves the creation and addition of decision trees sequentially, each attempting to correct the mistakes of the learners that came before it. This raises the question as to how many trees (weak learners or estimators) to configure in your gradient boosting model and how big each tree should be.

- Booster: type of model to run at each iteration. gbtree: tree-based model was used.
- Max_depth: the maximum depth of a tree. The value 6 gave reasonable results.
- n_estimators : 100 was used.



```
## Train XGboost Model

import xgboost as xgb
from sklearn.metrics import accuracy_score

scores_train = []
scores_test = []


xg = xgb.XGBClassifier(booster = 'gbtree', max_depth = 6, n_estimators=100)
xg.fit(X_train, y_train)

scores_train.append(xg.score(X_train,y_train))
scores_test.append(xg.score(X_test,y_test))

# Predict the response for test dataset

print("Average Accuracy on trainset:", round(np.mean(scores_train),2))
print('Average Accuracy on Test Data: ',round(np.mean(scores_test),2))

y_pred = xg.predict(X_test)
print("Accuracy:", round( metrics.accuracy_score(y_test, y_pred),2))
```

Figure 10 Classifying Data Using XGBoost

```
Average Accuracy on trainset: 1.0
Average Accuracy on Test Data:  0.5294117647058824
Accuracy: 0.5294117647058824
```

2.5 Evaluating

In the table below the accuracy result of each model implemented on the dataset (train – test).

Modeling Algorithm	Mean Accuracy on Training Set	Mean Accuracy on Testing Set	Deference between Accuracies
Random Forest	0.74	0.45	0.29
Decision Tree	0.83	0.43	0.4
SVC	1.0	0.47	0.53
KNN	0.59	0.45	0.14
XGBoost	1.0	0.53	0.47

- Those results show that in Decision Tree accuracy on the test set is low compared to other algorithms.
- SVC and XGBoost recorded the best results on test set, thought the difference between train set and test set is high, which indicate overfitting.
- XGBoost provided the best performance among all, therefore it will be adopted.
- The best accuracy on Test Data is 0.53.

3 Conclusion

To sum up, the best prediction model achieved 53% accuracy on the test, is XGBoost using 100 estimators, gbtree booster and max_depth = 6.
This accuracy is fairly low as the dataset is not rich in my opinion.