DATA INTENSITIVE COMPUTING PROJECT REPORT
PHASE – II
PREDICTING A NEW AIRBNB USER'S FIRST DESTINATION

mchava2 – Manish Chava
ssiddamr – Sai Sreekar Reddy Siddamreddy
Class – CSE 587

Problem Statement:

Airbnb is a multi-million-dollar company which lets user rent their home online. This company was founded in 2007 and ever since it has been a pioneer in the industry, being in the market for almost 2 decades, the company has so much data, which analyzed thoughtfully can provide very useful information to the company which can help them to sustain in the current competitive world.

The data which we have taken for this phase and the entire project is from Kaggle[1]. The data contains information about user's demographics, web sessions etc. (Columns have been explained below), using this data about the user we are trying to predict the new user's first destination. With this analysis or prediction, the company can develop their recommendation systems and can recommend homestays to a user in the predicted destination.

Steps followed before building model :

i) Data cleaning and preprocessing (which was done for phase I)
ii) The prediction would be done based on classification tasks, hence before passing the target value (country_destination) categorical variable as a string format it was preprocessed and turned into numerical value.

Models:

The models (along with their accuracy) which have been considered to predict are:

i) Gaussian Naïve Bayes Classifier – 0.559

ii) Random Forest Classifier – 0.601
iii) KNN Classifier – 0.608
iv) AdaBoost Classifier – 0.618
v) XgBoost Classifier – 0.633

GAUSSIAN NAÏVE BAYES CLASSIFIER:

Initially we started with Gaussian Naïve Bayes classifier model with the assumption that data is normally distributed, this isn't complex compared to the other classifiers and to see the prediction accuracy of this simple model, but the accuracy of this model wasn't satisfactory i.e; 0.559 (55%). Gaussian Naïve Bayes Classifier predicts by calculating the probability of each target class given with the feature values and concludes the prediction with the class which got highest probability value.

Reason(s) we have chosen this classifier is that the speed of this classifier, its simplicity and assumptions as our dataset contains more than 400,000 rows we have gone with this model initially. The other main factor which influenced us in selection of this classifier is that this classifier's scalability is too good, it can handle large datasets and high dimensionality data and the dataset which have considered has 12 dimensions. The reason behind this unsatisfactory accuracy might be because of the reason, data is not normally distributed.

Code:

Gaussian Naive Bayes

```
# Importing the required library to implement Gaussian Naive Bayes classifier
from sklearn.naive_bayes import GaussianNB
```

```
# Creating an object gaussianNBModel of class GaussianNB
gaussianNBModel = GaussianNB()
```

```
# Using the fit function of GaussianNB class the data has been fit
gaussianNBModel.fit(X_train,y_train)
```

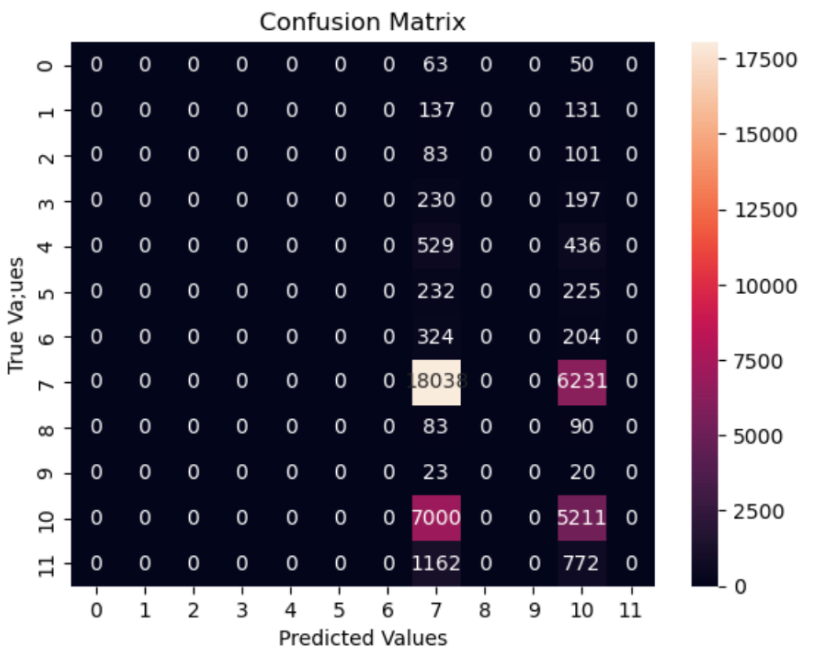
```
GaussianNB()
```

```
# Making predictions using the predict function of GaussianNB class on the X_test dataset
gnbPredictions = gaussianNBModel.predict(X_test)
```

```
print('The accuracy of Gaussian Naive Bayes Classifier is: ',accuracy_score(y_test,gnbPredictions))
```

```
The accuracy of Gaussian Naive Bayes Classifier is:  0.559246608294044
```

Heat Map:



Interpretation:

From the above Heat Map we can make the following intuitions:

1. For class 0 the correct predictions = 0
2. For class 1 the correct predictions = 0
3. For class 2 the correct predictions = 0
4. For class 3 the correct predictions = 0
5. For class 4 the correct predictions = 0
6. For class 5 the correct predictions = 0
7. For class 6 the correct predictions = 0
8. For class 7 the correct predictions = 18038
9. For class 8 the correct predictions = 0
10. For class 9 the correct predictions = 5211
11. For class 10 the correct predictions = 5567
12. For class 11 the correct predictions = 0

RANDOM FOREST CLASSIFIER:

Random forest classifier works by creating decision trees and predict the value by combining the result of their trees during training.
The second classifier which we have considered to predict is the Random Forest Classifier because Random Forest Classifier does not make any assumptions about the data and before making any predictions the classifier combines the outputs from the entire set of decision trees and assigns the one with the highest sum for a class. Random Forest Classifier also provides the insights of the features and their importance. The accuracy of this classifier is 0.601 (~60%) which is also not satisfactory, this might be because of the following reasons:

1. Random Forest has the capabilities to understand complex Data, but this sometimes comes at the cost of overfitting. This ultimately leads to less test accuracy.
2. Random Forest is not good with new data, forces to rebuild the trees. This leads to un-satisfactory results when dealing with new test data.

Code:

RANDOM FOREST CLASSIFIER

```
# Importing the needed library to implementing Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier
```

```
# Create an object of Random Forest Classifier class
randomForest = RandomForestClassifier(n_estimators = 100, random_state = 1)
```

```
# Using the object randomForest, the dataset is fitted using fit function
randomForest.fit(X_train,y_train)
```

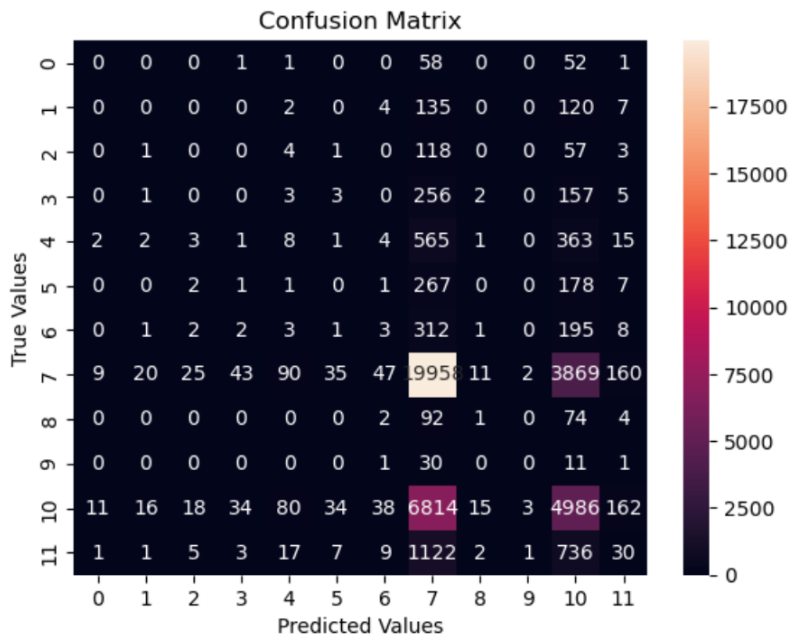
```
RandomForestClassifier(random_state=1)
```

```
# Using the predict function of Random Forest Classifier predictions are made on the X_test.
randomForestpred = randomForest.predict(X_test)
```

```
print('The accuracy of Random Forest Classifier is: ',accuracy_score(y_test,randomForestpred))
```

```
The accuracy of Random Forest Classifier is:  0.6010295391128644
```

Heat Map:



Interpretation:

From the above Heat Map we can make the following intuitions:

1. For class 0 the correct predictions = 0
2. For class 1 the correct predictions = 0
3. For class 2 the correct predictions = 0
4. For class 3 the correct predictions = 0
5. For class 4 the correct predictions = 8
6. For class 5 the correct predictions = 0
7. For class 6 the correct predictions = 3

8.For class 7 the correct predictions = 19958
9.For class 8 the correct predictions = 1
10 For class 9 the correct predictions = 0
11For class 10 the correct predictions = 4986
12For class 11 the correct predictions = 30

KNN CLASSIFIER:

The third classifier which we have considered to predict is the KNN Classifier because KNN is also very well at handling multi classes, and it does not assume that the data is normally distributed. KNN works by assigning the data point to the nearest k classes. The accuracy provided by this classifier is 0.608 (~60%), there is slight improvement in the accuracy compared to the Gaussian Naive Bayes, but the result is not so satisfactory. This might be because KNN is sensitive to the features which are not irrelevant features and yes, there are some features which have negative correlation.

Code:

KNN

```
# Importing the needed library to implementing KNN
from sklearn.neighbors import KNeighborsClassifier
```

```
# Creating an object of class KNeighborsClassifier with 12 neighbors,
#because in the target column there are 12 unique destinations
knnClassifier = KNeighborsClassifier(n_neighbors=12)
```

```
# using fit function from KNeighborsClassifier, the data have been fitted.
knnClassifier.fit(X_train,y_train)
```

```
KNeighborsClassifier(n_neighbors=12)
```

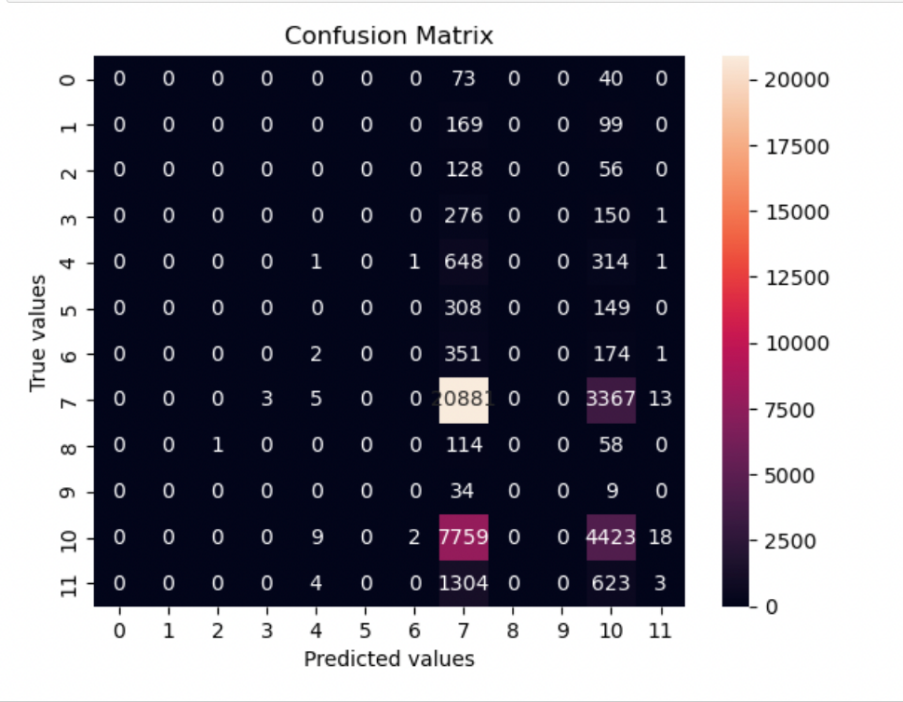
```
knnPredictions = knnClassifier.predict(X_test)
```

```
/Users/sreekarreddy/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_classification.py:228: FutureWarnin
g: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the
axis it acts along. In SciPy 1.11.0, this behavior will change: the default value of `keepdims` will become False, th
e `axis` over which the statistic is taken will be eliminated, and the value None will no longer be accepted. Set `ke
epdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

```
print('The accuracy of KNN Classifier is: ',accuracy_score(y_test,knnPredictions))
```

```
The accuracy of KNN Classifier is:  0.6087751371115173
```

Heat Map:



Interpretation:

From the above Heat Map we can make the following intuitions:

1For class 0 the correct predictions = 0
2.For class 1 the correct predictions = 0
3.For class 2 the correct predictions = 0
4.For class 3 the correct predictions = 0
5.For class 4 the correct predictions = 1
6.For class 5 the correct predictions = 0
7.For class 6 the correct predictions = 0
8.For class 7 the correct predictions = 20883
9.For class 8 the correct predictions = 0
10 For class 9 the correct predictions = 0
11For class 10 the correct predictions = 4423
12For class 11 the correct predictions = 3

AdaBoost Classifier:

The fourth classifier which we have chosen is AdaBoost Classifier. AdaBoost works by combining weak decision trees to make itself a strong classifier.

AdaBoost is also a ensemble learning technique which builds weak learners in sequential order where each weak learner adapts to the previous weak learner.

Adaboost is adaptive to the misclassifications in the data from each model.

Here, each model is influenced by the performance of the previous model. At each stage for each model only the misclassified data from the previous model is given and giving more significance to the misclassified data in the successive models. This can be a significant factor for contributing to higher accuracy than Random Forest and all the previous Classifiers.

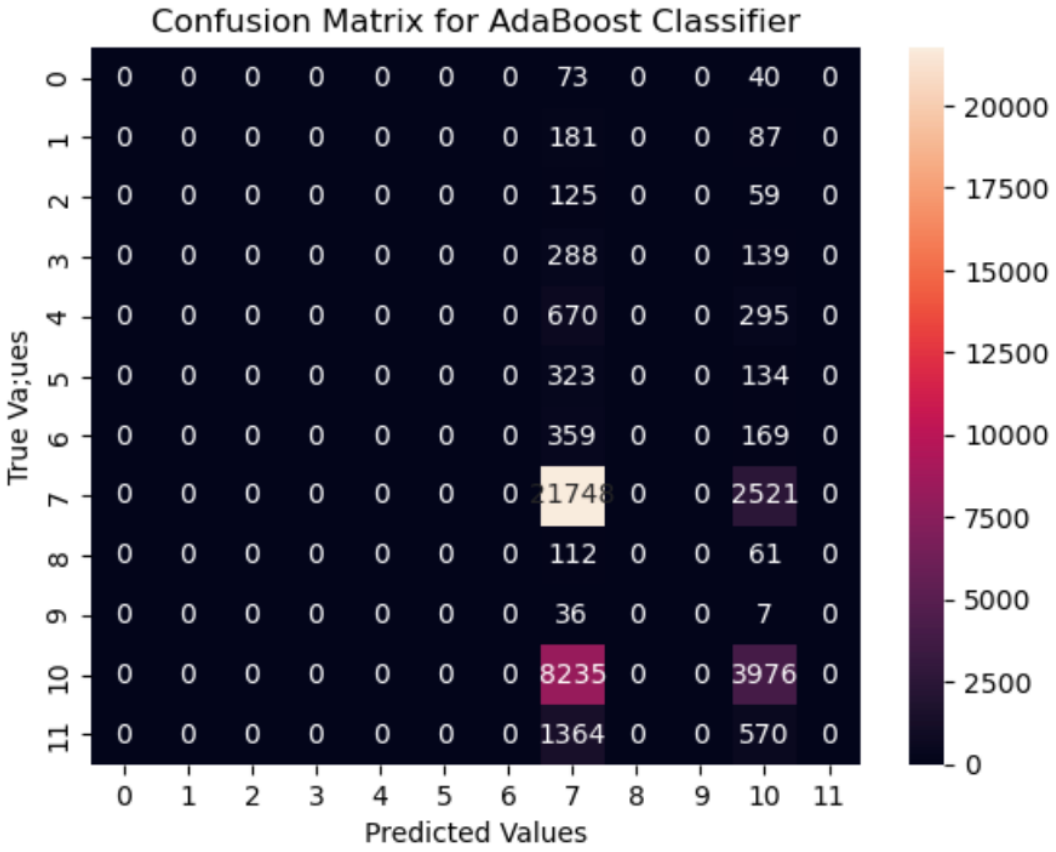
Code :

AdaBoost

```
# Importing the required library to implement AdaBoost classifier
from sklearn.ensemble import AdaBoostClassifier

# Creating an object adaBoostModel of AdaBoostClassifier
adaBoostModel = AdaBoostClassifier(n_estimators = 12)

# Using the fit function from AdaBoost function, data has been fitted
adaBoostModel.fit(X_train,y_train)

# Making predictions on X_test data using the predict function of the class AdaBoost
adaBoostPredictions = adaBoostModel.predict(X_test)

print('The accuracy of XGBoost Classifier is: ',accuracy_score(y_test,adaBoostPredictions))

sns.heatmap(confusion_matrix(y_test, adaBoostPredictions), annot=True, fmt='d')
plt.xlabel('Predicted Values')
plt.ylabel('True Va;ues')
plt.title('Confusion Matrix for AdaBoost Classifier')
plt.show()
```

```
The accuracy of XGBoost Classifier is:  0.6187818724141249
```

Heat Map:



Interpretation:

From the above Heat Map we can make the following intuitions:

1. For class 0 the correct predictions = 0
2. For class 1 the correct predictions = 0
3. For class 2 the correct predictions = 0
4. For class 3 the correct predictions = 0
5. For class 4 the correct predictions = 0
6. For class 5 the correct predictions = 0
7. For class 6 the correct predictions = 0
8. For class 7 the correct predictions = 21748
9. For class 8 the correct predictions = 0
10. For class 9 the correct predictions = 0

11For class 10 the correct predictions = 3976
12.For class 11 the correct predictions = 0

XgBoost Classifier:

The fifth classifier which we have chosen to predict the values is XgBoost Classifier. Similar to AdaBoost classifier, this classifier as well combines weak classifiers and creates a strong classifier the only difference is that this classifier uses the gradient boosting. As there is huge data and many features compared to the other classifiers XgBoost classifier seemed to capture them all efficiently and provided the better accuracy compared to the other classifiers. The accuracy provided by this classifier is 0.633 (~63%).

The Gradient Boosting algorithm used in XgBoost tends to find a faster route to the minimum value, so it converges to the global minimum faster with fewer steps. Thus it is giving the highest accuracy across all the five models.

Code:

XGBoost

```
# Importing the needed library to implement XGBoost
from xgboost import XGBClassifier
```

```
# Creating an object xgbModel of XBGClassifier
xgbModel = XGBClassifier()
```

```
# Using the fit function from XBGClassifier function, data has been fitted with default parameters.
xgbModel.fit(X_train,y_train)
```

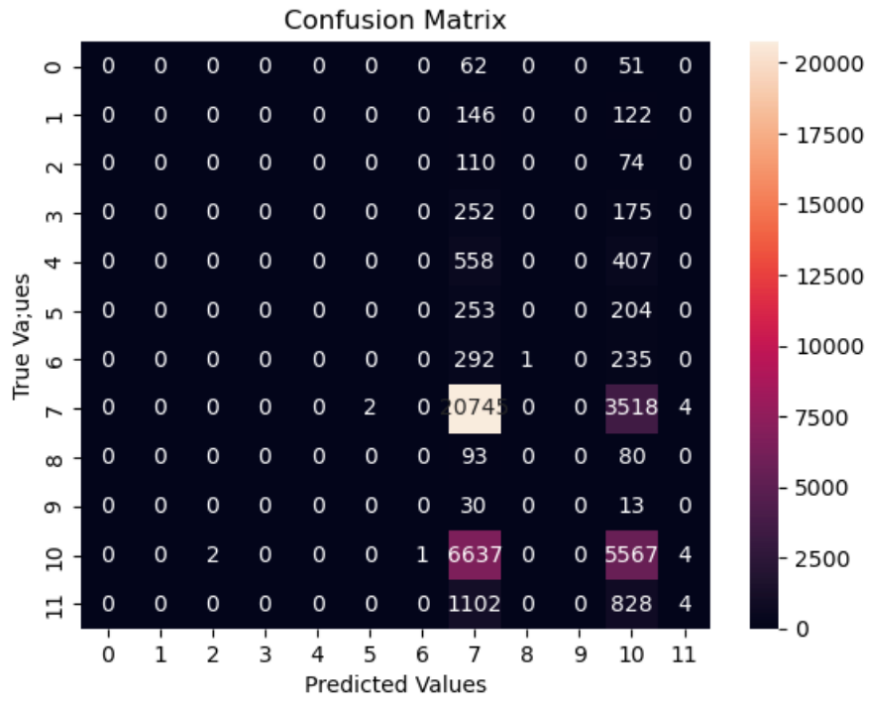
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,
              objective='multi:softprob', predictor=None, ...)
```

```
# Making predictions on X_test data using the predict function of the class XGBClassifier
xgbPredictions = xgbModel.predict(X_test)
```

```
print('The accuracy of XGBoost Classifier is: ',accuracy_score(y_test,xgbPredictions))
```

```
The accuracy of XGBoost Classifier is:  0.6330222264986048
```

Heat Map:



Interpretation:

From the above Heat Map we can make the following intuitions:

1. For class 0 the correct predictions = 0
2. For class 1 the correct predictions = 0
3. For class 2 the correct predictions = 0

4.For class 3 the correct predictions = 0
5.For class 4 the correct predictions = 0
6.For class 5 the correct predictions = 0
7.For class 6 the correct predictions = 0
8.For class 7 the correct predictions = 20745
9.For class 8 the correct predictions = 0
10. For class 9 the correct predictions = 0
12.For class 10 the correct predictions = 5567
13.For class 11 the correct predictions = 11

The reason for less correct predictions for classes 0 to 6 can be because of the highly unbalanced dataset. The data is highly populated with classes 7,10 and 11.

Hence the model was able to learn about these three classes but due to less data about classes 0 to 6 the model was not able to learn about these classes properly.

Dataset Link :
https://www.kaggle.com/competitions/airbnb-recruiting-new-user-bookings/overview

References :

https://machinelearningmastery.com/boosting-and-adaboost-for-machine-learning/
https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/
https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/#:~:text=Specificity%20tells%20us%20what%20proportion,correctly%20identified%20by%20the%20model
https://www.kaggle.com/general/218408
https://www.geeksforgeeks.org/xgboost/