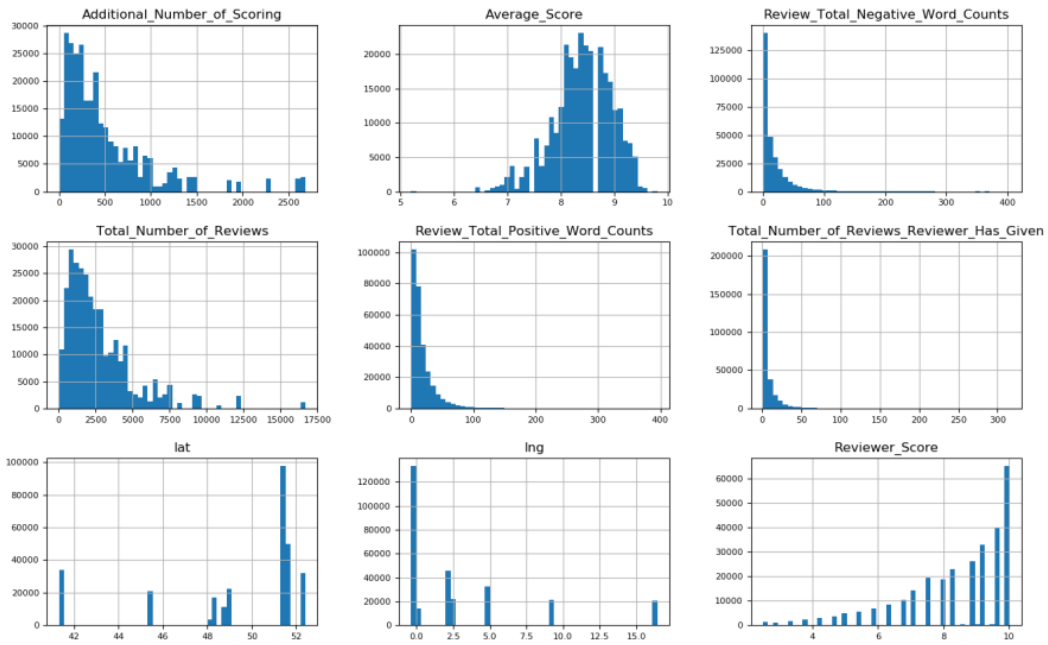
Hotel regressor

Team ID: SC_26

عبدالرحمن محمد زينهم	20201700445
ادهم محمد فاروق عواد	20201700101
عبدالرحمن موسى سعيد عبداللطيف	20201701101
يوسف محمد احمد السيد	20191700782
يوسف سيف النصر عمار عبداللطيف	20201701010

In this report, we will explore the problem of predicting hotel ratings using a regression model. Our goal is to develop a model that can accurately predict the overall rating of a hotel based on various features, such as the number of reviews, negative and positive comments, and the location. To do this, we will use a dataset of hotel reviews and ratings, and we will evaluate the performance of different regression algorithms using various metrics such as score and mean squared error.



Preprocessing

First, we started in the data cleaning process by dropping null values in our data by using pandas library we dropped exactly 1824 rows from our data set the null rows was in the Lat and lang columns

```
data.dropna(how='any',inplace=True)
```

```
Review_Total_Negative_Word_Counts                0
Total_Number_of_Reviews                          0
Positive_Review                                  0
Review_Total_Positive_Word_Counts                0
Total_Number_of_Reviews_Reviewer_Has_Given       0
Tags                                             0
days_since_review                                0
lat                                           1824
lng                                           1824
Reviewer_Score                                   0
```

we tried to replace the null values by the mean values but this process was so heavy and

The code complexity was high because the data have a total of 290000 rows.

Then in the next step we searched for duplicates values in our dataset we found 174 rows that are duplicated that's too little for the size of the data set so we dropped it using pandas library.

```
dublicates:
174
```

```
data=data.drop_duplicates()
```

Now we started to split the columns to X and Y. X for the Features and Y for the result column ('Reviewer Score').

```
Y=data['Reviewer_Score']
X=data.iloc[:,0:15]
```

After this it was the time to deal with the none numerical features we started with the hotel address column and make a new column from it that contains only the country name that the hotel from because there

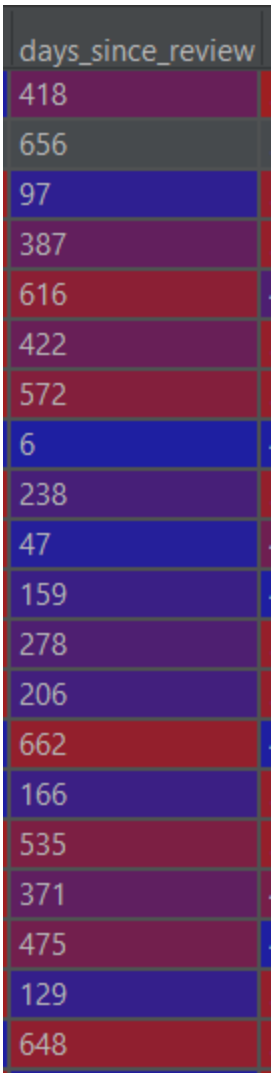
was missing data in the streets address and post codes in a lot of columns and that will damage the data set so we removed all the addresses and left the country of the hotel.

```
data['Hotel_Address'] = (data['Hotel_Address'].str.split(' ').str[-1]).str.lower()
```

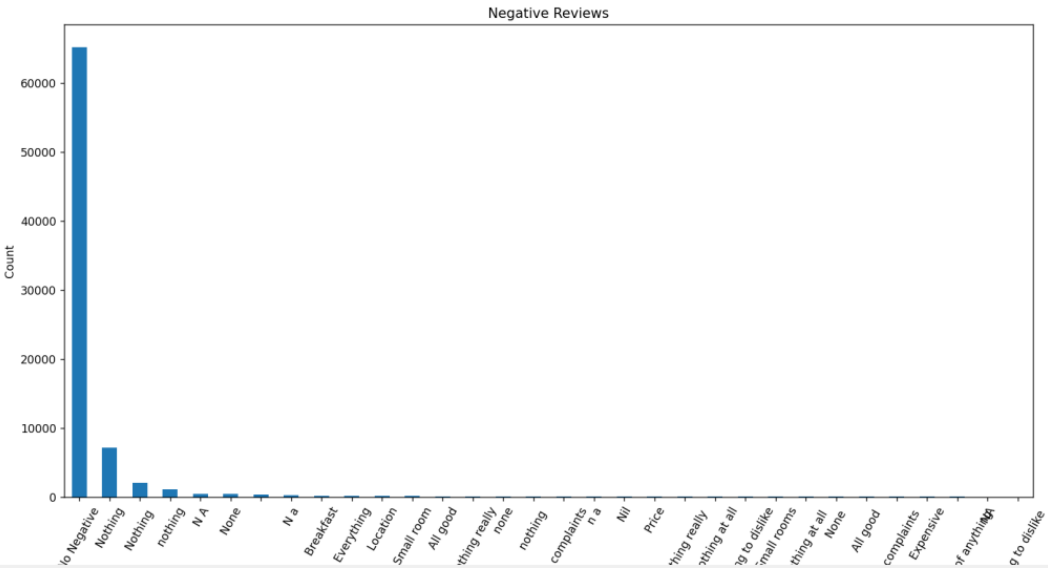
Then we started with the Review_date columns we split it into days, months, years columns then we saw the correlation of each of them with the Reviewer Score column and choose the one with highest correlation and made it the new date column and dropped the other two which was the months column.

Review_Date
5
3
8
7
5
7
2
10
4
7
4
10
3
7
2
8
9
3
6
8

The days since review column needed a split too to neglect the days word and make it only a numerical value to make it logical.



Some columns needed NLP to deal with it like the Negative review and positive review so we choose the most repeated words in this column by using data analysis graphs that pointed that 70% of the rows have the words Nothing ,nothing and No Negative on them so we put this words in a list then we looped at the column of Negative Review and Positive Review so we make all to lower case so we can find our words from the list then if we find this three words in the Negative review we put zero else we put one and the opposite in the Positive one .



Then we started with encoding the features that have category values we put them all in a list that named cols

```
cols=('Reviewer_Nationality','Hotel_Name','Hotel_Address','Tags','Positive_Review','Negative_Review')
```

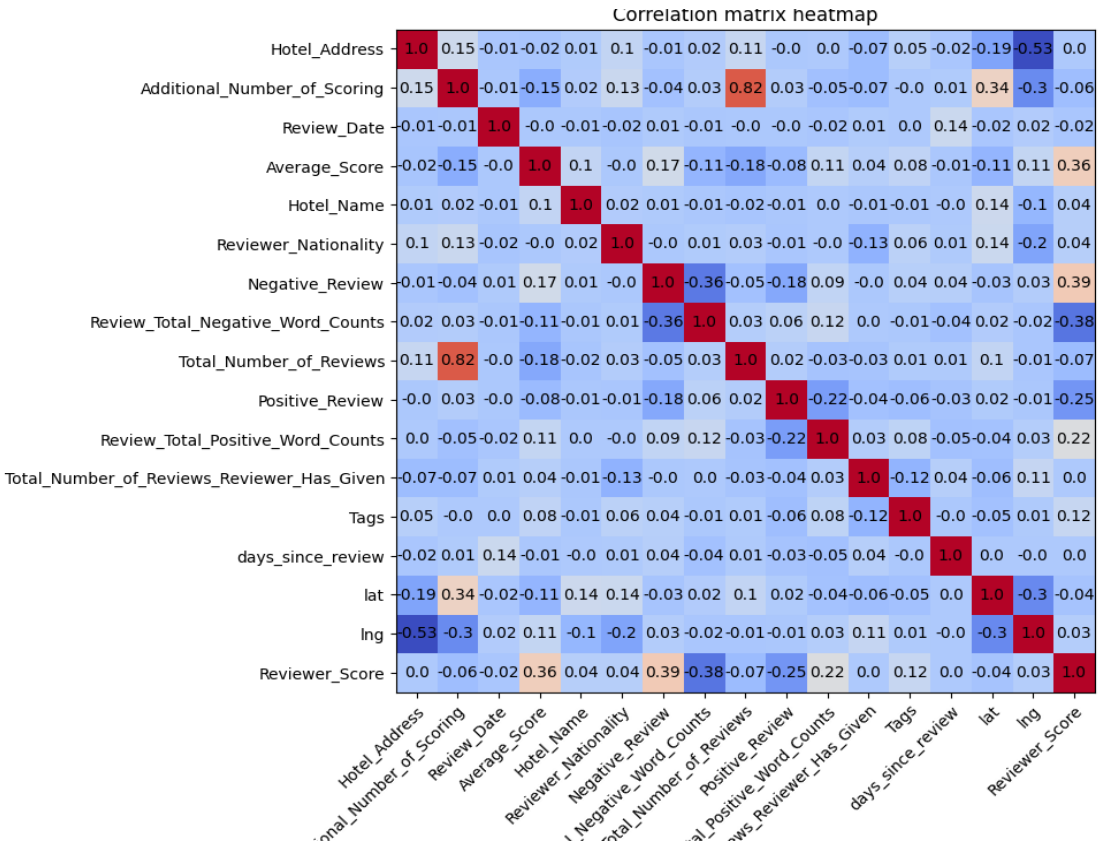
Then we build a function named Feature Encoder we passed the data set and the cols list to it this function is used to take more than one feature and encode them so we do not need to make a lot of lines to make encoding for every feature this function loops in the feature that was passed to it then pass it to the function Label Encoder from SK_Learn library to fit the data on it then it return all the feature that was passed to it encoded.

```
def Feature_Enconder(X, cols):
    for c in cols:
        lbl = LabelEncoder()
        lbl.fit(list(X[c].values))
        X[c] = lbl.transform(list(X[c].values))
    return X
data=Feature_Enconder(data,cols)
```

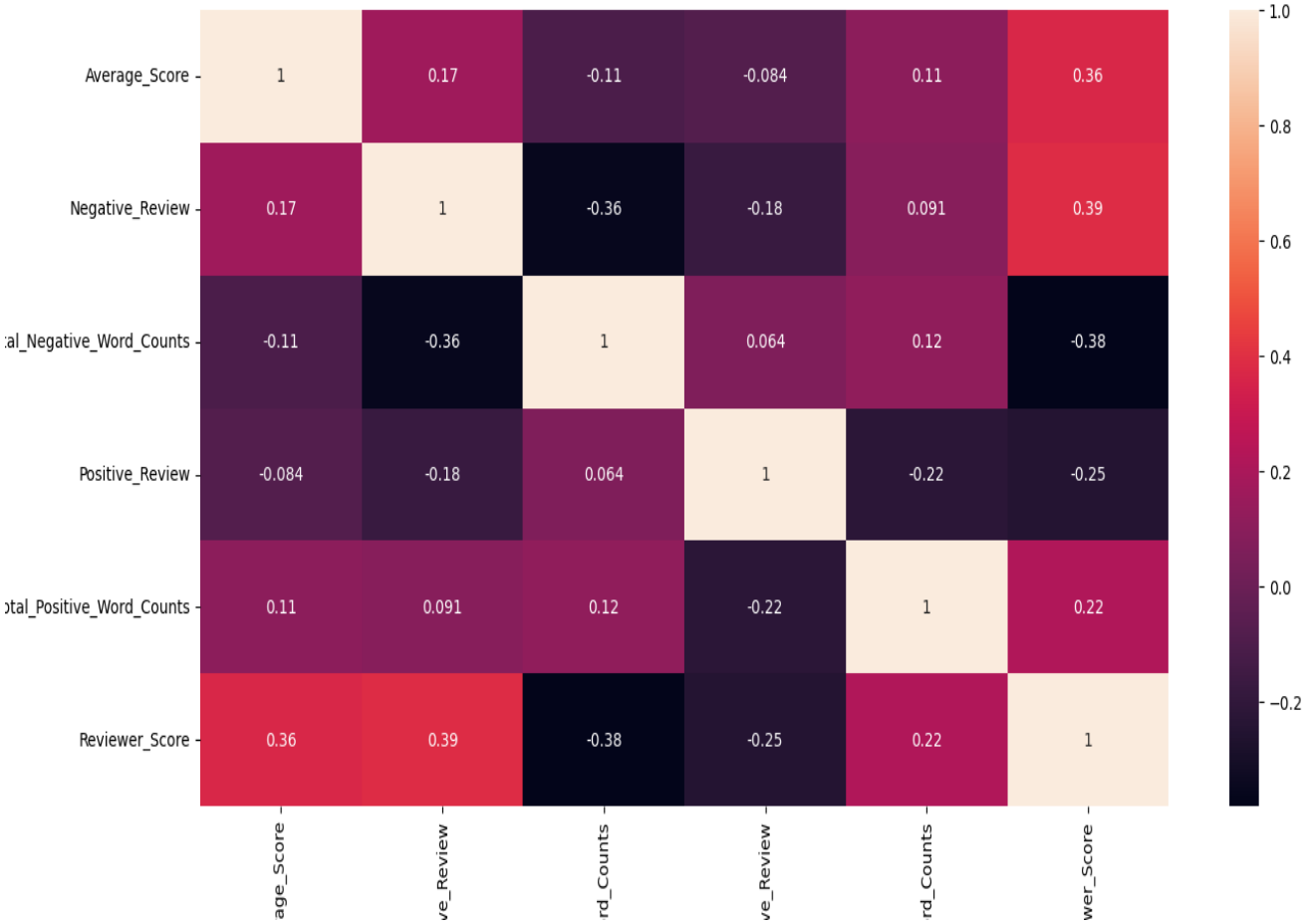
We needed to scale our data because we are using a random forest algorithm which is a distance-based algorithm we had a lot of options like min max, standard scaler and Z-score normalization we choose the Z-score normalization because it handles outliers well. Although it does not normalize the data with the same scale as the min max but we have a lot of outliers so we needed a to scale with the z score to handle them.

We also tried to use polynomial features with degree 4 because more than that the error was getting higher, and it was over fitting on the training data the polynomial features give better score of course than the normal feature on the linear regression model.

And now we come to the last stage of the data preprocessing the feature selecting part for this one we made a correlation matrix to see the correlation between every feature and the reviewer score we then choose every feature that have more than 0.2 correlation score with it



then we draw another heatmap to show the remaining feature we choose the 0.2 because it's the one that gives the best score with our used algorithms we absolutely was going to drop columns like Lat and Ing because they got no relation with the review but the correlation was so low any way so it got dropped automatically so at the end there was only 5 features out of 16 features that had a high correlation with the reviewer score and this features are (Average Score – Negative review – Review total negative words count -Positive review -review total positive words count) and we dropped the other 10 features.



In the split we first tried to make it with K-fold to split the data set in an accurate way at the expense of the complexity of the code as the code now have a very high complexity because in k fold we tried to split the data set 10 times to get the best score and to get the best model of them but when we find that this makes the code run slow because of the complexity.

```
# cv = KFold(n_splits=5, random_state=1, shuffle=True)
#
# # Build multiple linear regression model
# model = MLPRegressor()
#
# # Use k-fold CV to evaluate model
# scores = cross_val_score(model,X_train ,y_train , scoring='neg_mean_absolute_error', cv=cv, n_jobs=-1)
#
# # View mean absolute error
# mean_absolute_error = abs(scores.mean())
```

so We split the data set to 80% train and 20% test then we split the train test to 10% validation 90% train we found that this is the best splitting we can get that fits our data set and gives the best score

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.20,shuffle=True,random_state=10)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.10,shuffle=True, random_state=42)
```

we then trained our modules then fitted them on the validation data set to see the best model of them we compared them by the score and mean square error of each we found that the best one of them that gives the lowest error is the random forest so we can try the test case on it and as expected it gives a great prediction.

We used four regression techniques to predict the person review like linear regression, Random Forest, neural network regressor and SVM.

Some of the advantages of using random forest include:

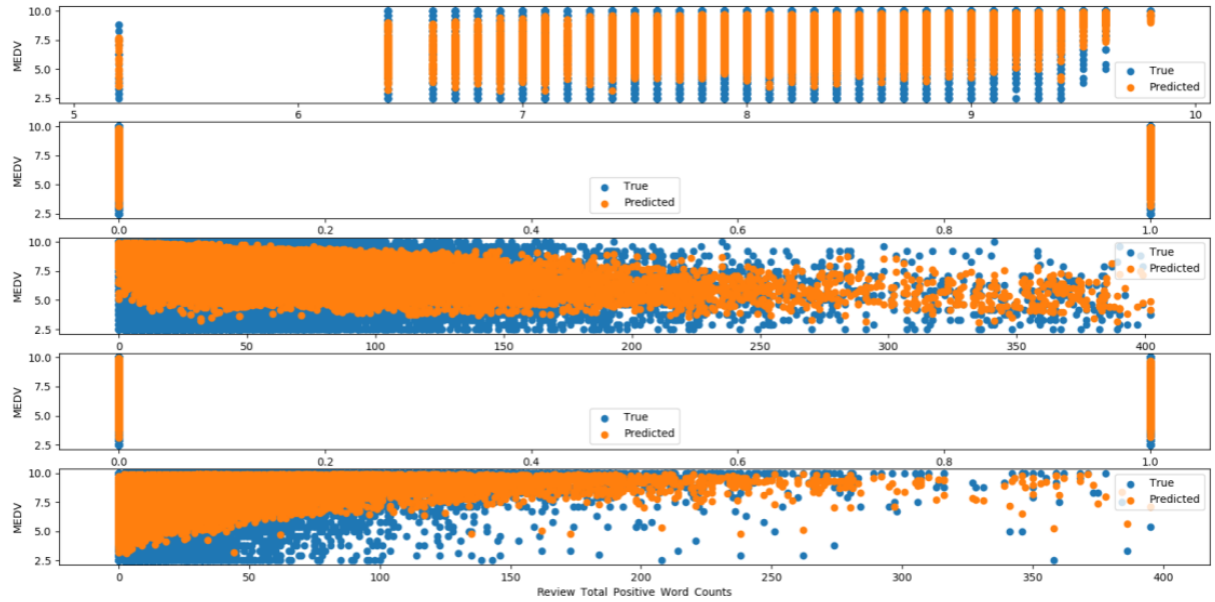
It is a very accurate algorithm.

It is very robust to overfitting.

It can be used to solve a wide variety of problems.

It is relatively easy to understand and implement.

And here its resultant(s) regression line plot



Some of the disadvantages of using random forest include:

It can be computationally expensive to train and, that really happened with us it takes a lot of time.

It can be sensitive to the choice of hyperparameters.

It can be difficult to interpret the results.

But after all Random Forest is a very powerful algorithm that can be used to solve a wide variety of problems.

And her its evaluation from our code:

```
Mean Squared Error by using RandomForestRegressor: 1.4595287041328897
r2_Score using RandomForest  0.44234449362857886
True value for the first Reviewer_Score : 8.8
Predicted value for the first Reviewer_Score : 9.161431839892623
```

Here are some of the advantages of using neural networks:

They are very powerful and can be used to solve a wide variety of problems.

They can learn from data and improve over time.

They are very good at identifying patterns in data.

Here are some of the disadvantages of using neural networks:

They can be computationally expensive to train it takes like 2 minutes to run .

They can be difficult to interpret.

They can be sensitive to the choice of hyperparameters.

Neural networks are a powerful tool that can be used to solve a wide variety of problems and it gives the best mean square error so it is worth to use even of this disadvantages.

And her its evaluation from our code on the validation data set:

```
Mean Squared Error by using NeuralNetworkRegressorModel: 1.4555002578404483 mean
r2_Score 0.4438836790181777
True value for the first Reviewer_Score by using NeuralNetworkRegressorModel: 8.8
Predicted value for the first Reviewer_Score by using NeuralNetworkRegressorModel: 9.14763298858679
```

And this on the test data set:

```
Mean Squared Error by using the best:  1.5208808193370837
r2_Score 0.4351758849917371
Mean Absolute Percentage Error:  13.361775906071108
True value for the first Reviewer_Score by using the best model : 8.8
Predicted value for the first Reviewer_Score by using the best model: 9.006131181749465
```

And here some of the other algorithms that we used and their evaluation:

Normal linear regression:

```
Mean Square Error 1.6605758268389568
r2_Score 0.3655285771620528
```

regression with polynomial features:

```
Mean Square Error by using PolynomialFeatures  1.4990119054061615
r2_Score 0.4272587851139992
True value for the first Reviewer_Score : 8.8
Predicted value for the first Reviewer_Score : 9.279607892036438
```

And finally we used voting technique to improve the accuracy robustness of prediction by combining all the algorithms that we used earlier on the project and here what we got from it:

```
VotingRegressor R^2 score: 0.43096784478722694
the voting mean square error is  1.5322116521821814
True value for the first Reviewer_Score by using voting: 8.8
Predicted value for the first Reviewer_Score by using voting: 8.797389891678685
```

Conclusion:

We actually before starting in this project we thought that it was going to be easy like AI project from last year but the data was challenging to deal with in a good way we learned a lot by watching the lecture's asking TA and of course searching it engorges us to learn in NLP too at some point for the bonus the problem that faced us that the data set of the hotel is so hard to deal with without a good knowledge in NLP that takes a lot of time I made a mistake at the biggening of the project I split the team 3 for the preprocessing part and 2 for the algorithms but the preprocessing found to be way harder with NLP and a lot of hard features that made the project take a lot of time but we managed to deal with it as hard as we could and we want to thank the TA for their useful support sections and Apple ai too it was really useful project we worked as a team and we all gained a good start for a very interesting field like machine learning.