Report :

(word count:: 1417)

After analyzing the data and using numerous classification and regression models, the hotel's profit is forecasted. The methodologies used in both Methods are detailed below.

First, I imported the libraries 'panda', 'Numpy', 'matplotlib', 'pyplot', and 'seaborn' into a jupyter notebook. These libraries serve the following purposes:

- Panda: is commonly used for data analysis and machine learning.
- Numpy: is used to perform many types of mathematical operations on arrays and matrices.
- Matplotlib: is a plotting library that is used for creating static, animated, and interactive visualization and graphics.
- Seaborn: is the same library as matplotlib but it is more comfortable in handling the data frame of pandas.
- Sklearn. processing: is a package that provides many usual utilities functionality and transformer classes to change the rough feature vectors into such representation which is more suitable for the downstream estimators in machine learning tasks.

1. Classification

• Data Cleaning:

After importing the data and reading the CSV files, the initial step will be to check for null values. The F21 column of the CSV file CE802 P2 Data.csv had 500 Null entries. Because there are 1000 rows altogether, 500 null values out of 1000 is a substantial quantity. The first option for dealing with null values is to simply remove them; however, when we assessed the data, we found that the accuracy was lower since removing the null values might affect the accuracy. As a consequence, we tried replacing null values with the column's mean in a different approach.

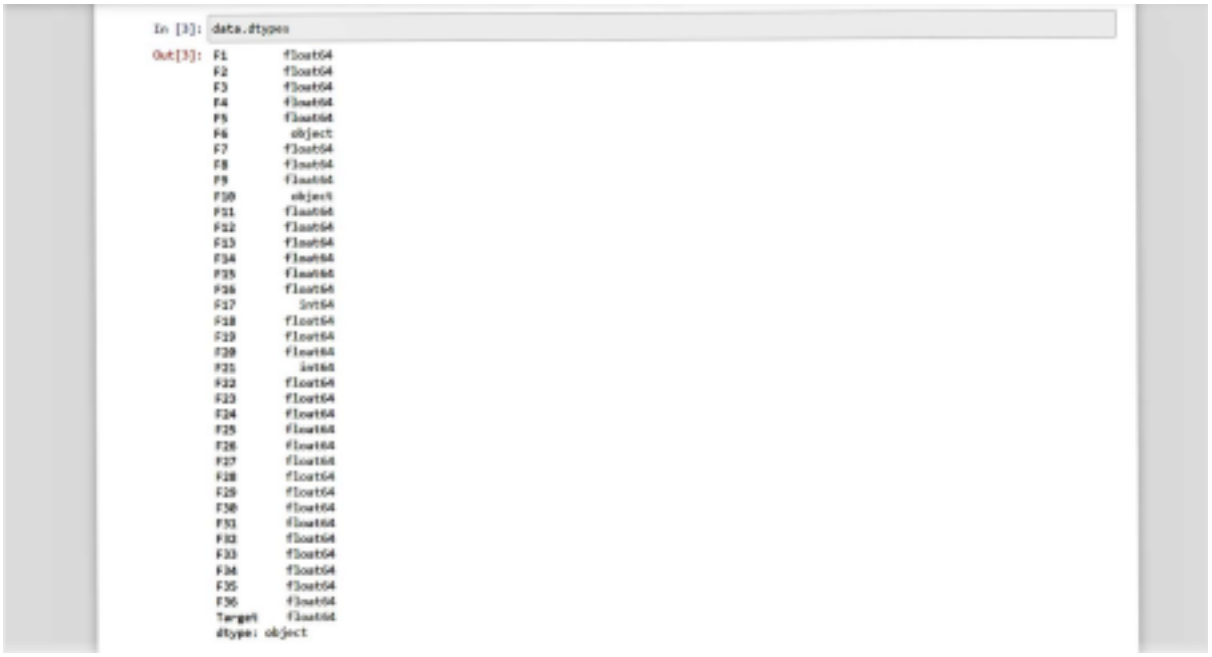
data ['F21'].fillna

data ['F21'].

mean (), inplace= True

The correctness of the model is discussed in further depth in the report. The column's data types were also checked to verify if they were numerical. Machine learning models that integrate non-numerical or

categorical input become unstable and fail to generate the desired outcomes.



This is a snapshot of the variables in the data and their data types.

The total of null values equals 0 when null values are substituted by the mean.

The Confusion Matrix of Decision Tree

```
<built-in function print>
[17] print(confusion_matrix(y_test,predictions))

     [[84 19]
      [25 72]]

[18] print(classification_report(y_test,predictions))

                   precision    recall  f1-score   support

            False       0.77      0.82      0.79       103
             True       0.79      0.74      0.77        97

         accuracy                           0.78       200
        macro avg       0.78      0.78      0.78       200
     weighted avg       0.78      0.78      0.78       200
```

• Scaling

After that, I clean up the data and remove the "class" column from the train data (P2 data), because the last column's data type is a string, and we can't work with strings.

After that, I perform a data normalization. To standardize the data, I utilized a feature scaling approach. The technique of feature scaling is used to normalize the range of independent variables or data components. As previously stated, we used Decision Tree, Random Forest, and Logistics models. Because the data points may differ from the average point, MinMaxScaler was employed to bring them closer together. The sklearn package was used to import MinMaxScaler.

There are several forms of feature scaling normalization, such as min-max normalization, mean normalization, and Z-score normalization (standardization). On both data sets, I used the "min-max scaler" for normalization. In min-max scaling, we subtract the dataset's minimum value from all of the values, then divide it by the dataset's range. In this situation, our dataset will always be between 0 and 1, but it will also be between -1 and +1 in certain circumstances.

Import MinMaxScaler from sklearn.preprocessing. To separate the data into two portions, we used a train test split once more.

Then I perform the data cleaning techniques. I replace all the missing values in column 'F21' in both data sets by the median of that column.

Then I worked on outliers in both data sets. "Outliers are data-items that differ dramatically from the rest of the objects. Errors in measurement or execution might cause them. Outlier mining is the term used to describe the analysis used to find outliers". First of all, I detect the outlier in the data by plotting the graph. For this purpose, I used the "plt. boxplot" function and draw the graph for one graph and it indicates the outliers are present.

• Models

Three distinct types of categorization models were used to predict. They were applied to two separate datasets. Null values are present in the first, but not in the second. Null values resulted in worse prediction accuracy, as we observed. The null values were replaced by the mean in the alternative, which resulted in a greater degree of accuracy.

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

     Accuracy: 0.79
```

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

Accuracy: 0.78
```

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

     Accuracy: 0.655
```

Accuracy of Decision tree: 0.78
Accuracy of KNN: 0.655
Accuracy of Random Forest: 0.79

The accuracies for the specified model are shown above, and we can see that the Random Forest is the most accurate. We can utilize Random Forest for further prediction because it has high accuracy. Random Forest needs less effort for data preparation during pre-processing than other methods. A Random Forest does not necessitate data normalization.

A Random Forest does not necessitate data scalability. In addition, missing values in the data have no significant impact on the Random Forest building process. A Random Forest model is simple to

understand and communicate to technical teams and stakeholders.

Unlike typical data pre-processing stages, the pre-processing phases in a Random Forest making model involve less coding and analysis.

Unlike typical data pre-processing stages, the pre-processing steps in a Random Forest making model save time. In a Random Forest , comprehensive rules are constructed.

In comparison to other algorithms, the concept that drives the Random Forest -making model is more familiar and easy for programmers.

One of the quickest ways to identify links between variables and the most significant variable is to use a Random Forest.

For better target variable prediction, new features can be added.

Outliers and missing values have little effect on Random Forest, and they can handle both numerical and categorical variables.

It makes no assumptions about space distributions or classifier construction because it is a non-parametric technique.

2. Regression

• Data Preprocessing:

We checked for the data type of values in each row and found that F5, F12, F21 & F28 were of different data types so we converted them into a float like others. This column would have thrown off the rest of the prediction, therefore it was converted to the appropriate form.

• Scaling:

For regression, the same procedures that were used for classification will be applied. MinMaxScaler was used to scale the data, and train test split was used to divide it into training and testing.

• Models:

Linear Regression- We discovered the data to be linear, hence Linear Regression was used. The regression method was used. The RMSE came out to be `934.6601256780118`, which was a nice match. Further, We discovered that when we used Linear Regression, the target variable became messed up.

We tried utilizing various models for additional prediction.

```
[48] print('MAE ',metrics.mean_absolute_error(y_test,predictions))
     print('MSE ',metrics.mean_squared_error(y_test,predictions))
     print('RMSE ',np.sqrt(metrics.mean_squared_error(y_test,predictions)))
```

```
MAE  757.1608832009081
MSE  873589.5505324369
RMSE  934.6601256780118
```

```
[49] metrics.explained_variance_score(y_test,predictions)
```

```
0.3694158470260557
```

```
[53] print('MAE ',metrics.mean_absolute_error(y_test,predictions))
     print('MSE ',metrics.mean_squared_error(y_test,predictions))
     print('RMSE ',np.sqrt(metrics.mean_squared_error(y_test,predictions)))
```

```
MAE  233.40850777777771
MSE  90984.91340194686
RMSE  301.6370557506933
```

```
[54] metrics.explained_variance_score(y_test,predictions)
```

```
0.9343349079403704
```

```
[58] print('MAE ',metrics.mean_absolute_error(y_test,predictions))
     print('MSE ',metrics.mean_squared_error(y_test,predictions))
     print('RMSE ',np.sqrt(metrics.mean_squared_error(y_test,predictions)))
```

```
MAE  731.7115390578078
MSE  818711.5166199266
RMSE  904.8267881865162
```

```
[59] metrics.explained_variance_score(y_test,predictions)
```

```
0.4087806303342061
```

Linear Regression Score Is `0.3694`

We also used Lasso which gave us similar scores as Linear Regression with an RSME of 904.82 and an accuracy of 0.4087.

Random Forest Regressor- To improve the error metrics we used nonlinear models.

The RMSE for Random Forest Regressor was 301.63, which was bothersome even if the RMSE was good.

Random Forest Regressor score is 0.93, which is the highest.

Random Forest Regressor- Among all the models, this one was the best because the RMSE for Random Forest Regressor was 0.03 in every aspect, which is ideal for future use.

Predictions of the value of the Target variable were excellent. Random Forest Regressor score 0.95

A random forest is a collection of decision trees that can be modeled for behavior analysis and prediction. In a forest, the decision tree cannot be trimmed for sampling or prediction selection. Because of its ability to operate with hundreds of variables, the random forest technique can manage big data sets. For additional prediction, use a random forest.

Random forest's adaptability is one of its most appealing features. It may be used for both regression and classification tasks, and the relative priority it assigns to the input features can be easily viewed. Random forest is also a useful method since the default hyperparameters it employs frequently produce accurate predictions. Understanding the hyperparameters is simple, and there aren't many of them to begin with. Because this is a regression problem, the target variable is numeric data that is predicted, thus we used the inverse transform function to descale it.

Because the preceding graph shows that random forest has the lowest error metrics, we used it.