Mid-Program Project -1

PG Program in AI & ML –NITW

Problem Statement:

Cab booking system is the process where renting a cab is automated through an app throughout a city. Using this app, people can book a cab from one location to another location. Being a cab booking app company, exploiting the understanding of cab supply and demand could increase the efficiency of their service and enhance user experience by minimizing waiting time.

Objective of this project is to combine historical usage pattern along with the open data sources like weather data to forecast cab booking demand in a city.

Dataset and Process flow:

Dataset and Process flow provided by Edureka. Please refer the problem statement document.

Project Code:

Various tasks were performed, analysis was done and the code is attached with the report.

Analysis and Inference:

1. Load Data – Train Data & Test Data:

The train data and test data are initially read from 'Train.csv' and 'Test.csv' files. Target variable "Total_Booking" is read from 'test_label.csv' and 'train_label.csv' and appended into train_data and test_data.

Train_data:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	Total_Booking
0	5/2/2012 19:00	Summer	0	1	Clear + Few clouds	22.14	25.760	77	16.9979	504
1	9/5/2012 4:00	Fall	0	1	Clear + Few clouds	28.70	33.335	79	19.0012	5
2	1/13/2011 9:00	Spring	0	1	Clear + Few clouds	5.74	6.060	50	22.0028	139
3	11/18/2011 16:00	Winter	0	1	Clear + Few clouds	13.94	16.665	29	8.9981	209
4	9/13/2011 13:00	Fall	0	1	Clear + Few clouds	30.34	33.335	51	19.0012	184

Size of train data: (8708, 10)

Test_data:

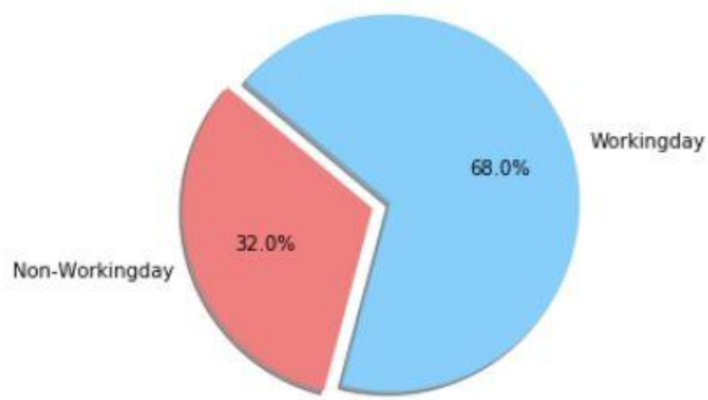
	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	Total_Booking
0	5/10/2012 11:00	Summer	0	1	Clear + Few clouds	21.32	25.000	48	35.0008	256
1	6/9/2012 7:00	Summer	0	0	Clear + Few clouds	23.78	27.275	64	7.0015	87
2	3/6/2011 20:00	Spring	0	0	Light Snow, Light Rain	11.48	12.120	100	27.9993	11
3	10/13/2011 11:00	Winter	0	1	Mist + Cloudy	25.42	28.790	83	0.0000	84
4	6/2/2012 12:00	Summer	0	0	Clear + Few clouds	25.42	31.060	43	23.9994	668

Size of test data : (2178, 10)

2. Task 1: (1.) <u>Visualize data and generate insights:</u>

Two plots were generated. The plots with their inferences have been explained below.

A) <u>Pie chart to understand Relationship between Working day and Total Booking</u>

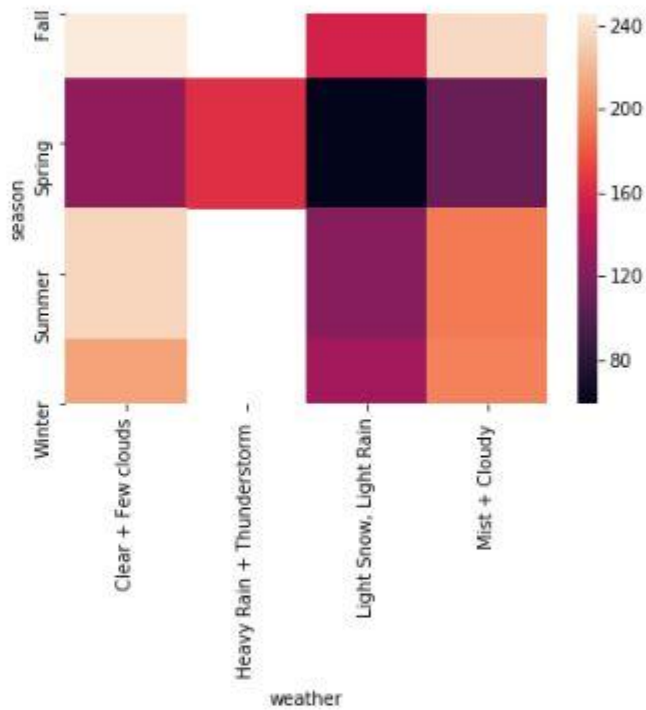


Out[5]:

col_0	count by workingday
workingday	
0	2784
1	5924

<u>Inference</u>: From the above Pie chart it is evident that the Cab demand is about 36% more on a Working Day compared to a Non -Working day

B) Heat Map to understand the influence of Seasons and Weather on the Average Booking



Inference: From the above heatmap, we understand that the average cab booking is the highest during 'Summer' and 'Fall' when the weather is 'Clear with Few clouds'. It goes high even when the weather is 'Misty and cloudy' during 'Fall'.

3. Task 1: (2.) Outlier Analysis :

First Label Encoding of the categorical columns 'Season' and 'weather' was done for both train and test data. Then the datetime column was converted from "hourly date +timestamp" to ordinal values. The resulting dataframe is as shown.

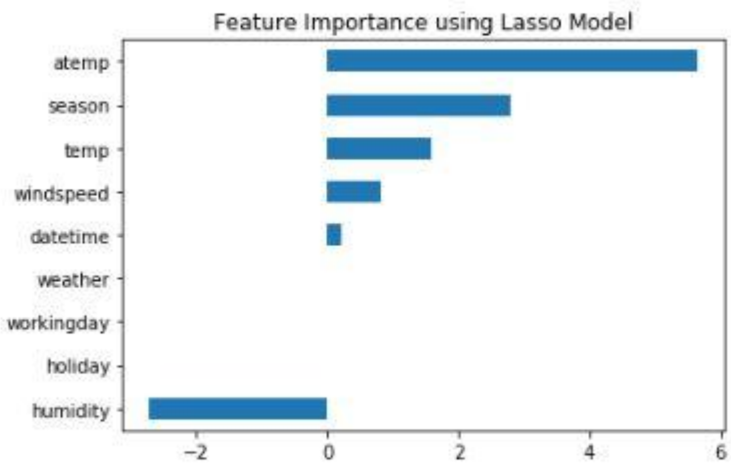
Train_data

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	Total_Booking
0	719163	2	0	1	0	22.14	25.760	77	16.9979	504
1	719163	0	0	1	0	28.70	33.335	79	19.0012	5
2	719163	1	0	1	0	5.74	6.060	50	22.0028	139
3	719163	3	0	1	0	13.94	16.665	29	8.9981	209
4	719163	0	0	1	0	30.34	33.335	51	19.0012	184

Test_data

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	Total_Booking
0	719163	2	0	1	0	21.32	25.000	48	35.0008	256
1	719163	2	0	0	0	23.78	27.275	64	7.0015	87
2	719163	1	0	0	1	11.48	12.120	100	27.9993	11
3	719163	3	0	1	2	25.42	28.790	83	0.0000	84
4	719163	2	0	0	0	25.42	31.060	43	23.9994	668

Before beginning the outlier analysis Feature Selection using Embedded Lasso model is also performed. This helped me to rule out few columns.

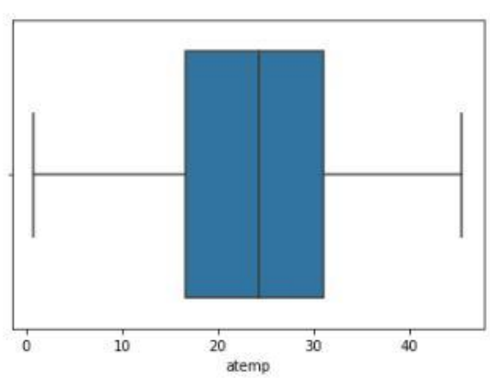


From the above Feature importance analysis we can eliminate 'Working Day', 'Weather' and 'Holiday' for outlier analysis since their coefficients are zero. Also 'Season' is excluded since it being Label Encoded and therefore will not have any outliers.

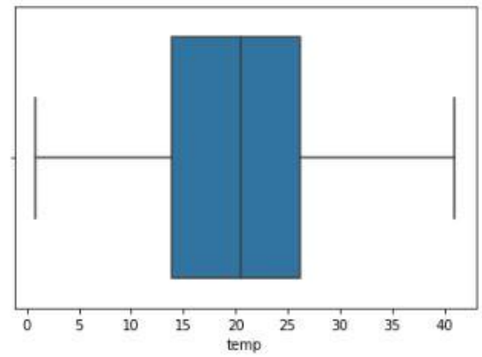
Outlier Analysis was done using two methods.

Method 1: <u>IQR based Method (Box Plot):</u>

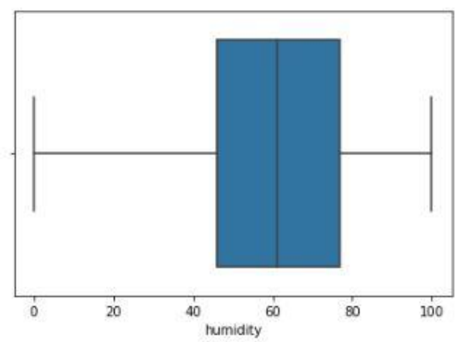
1. Box Plot for 'atemp' column



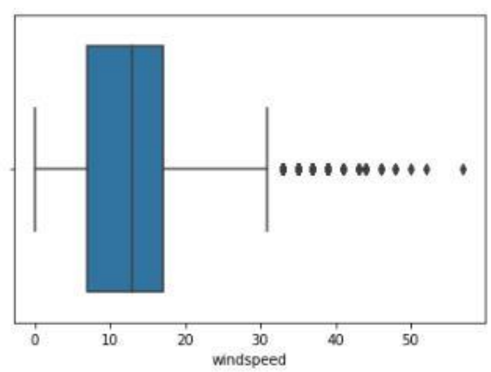
2. Box Plot for 'temp' column



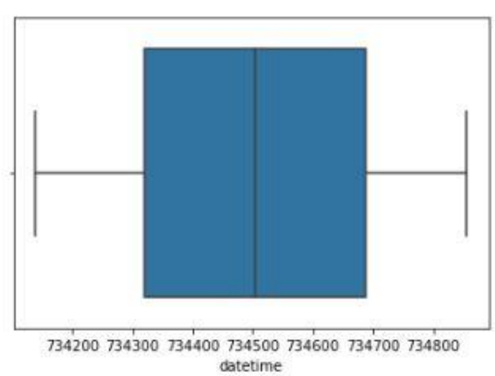
3. Box Plot for 'humidity' column



4. Box Plot for 'windspeed' column



5. Box Plot for 'datetime' colum



From the above analysis we see there are outliers only in the ‘windspeed’ column. Therefore, Z-score method is only applied for ‘windspeed’ column

Method 2: **Z-score Method:**
Obtained the number of outliers in the 'windspeed' column as **112.**

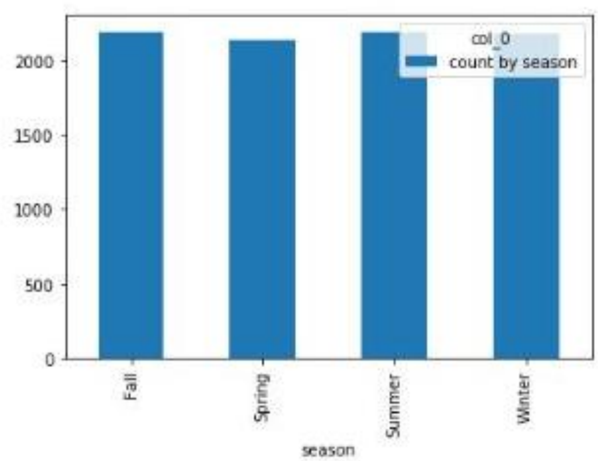
4. Task 1: (3.) Missing value analysis :

Presence of missing values in the data was checked using two methods. Iniatially a isnull() was used which returned a False for all columns. Then describe() returned the count value of all columns same as the totl entries in the data(8708), which indicated the absence of missing values.

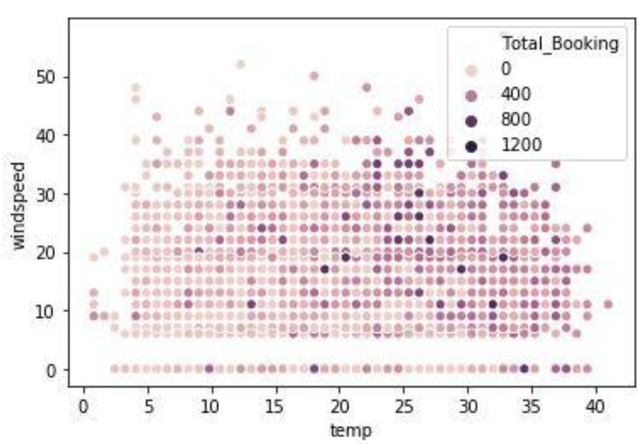
5. Task 1: (4.) Visualizing Total_Booking Vs other features:

a) Bar Plot to understand season-wise Total_booking

col_0	count by season
season	
Fall	2189
Spring	2136
Summer	2197
Winter	2186



b) Bar Plot to understand season-wise Total_booking



Though not quite evident, we can see a trend wherein when the temperature is in the range 20-35 and windspeed in the range 20-40 the Total booking tends to be around 800 or more.

6. Task 1: (5.) Correlation analysis :

The results of feature selection performed earlier were considered. In addition pearson correlation method was also performed.

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	Total_Booking
datetime	1.000000	0.192286	0.010277	0.006173	0.016994	0.183744	0.184282	0.034716	0.092120	0.314940
season	0.192286	1.000000	0.007236	0.005784	0.060624	0.380153	0.346962	0.061593	0.006530	0.008503
holiday	0.010277	0.007236	1.000000	0.249755	0.004602	0.000165	0.005526	0.004567	0.008075	0.004391
workingday	0.006173	0.005784	0.249755	1.000000	0.015095	0.032189	0.026168	0.009282	0.013035	0.012285
weather	0.016994	0.060624	0.004602	0.015095	1.000000	0.058179	0.053863	0.336430	0.028814	0.082382
temp	0.183744	0.380153	0.000165	0.032189	0.058179	1.000000	0.984035	0.066419	0.027824	0.397456
atemp	0.184282	0.346962	0.005526	0.026168	0.053863	0.984035	1.000000	0.044206	0.068911	0.392754
humidity	0.034716	0.061593	0.004567	0.009282	0.336430	0.066419	0.044206	1.000000	0.320346	0.307982
windspeed	0.092120	0.006530	0.008075	0.013035	0.028814	0.027824	0.068911	0.320346	1.000000	0.092090
Total_Booking	0.314940	0.008503	0.004391	0.012285	0.082382	0.397456	0.392754	0.307982	0.092090	1.000000

A threshold of 0.08 was then applied. It gave the following result.

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	Total_Booking
datetime	True	True	False	False	False	True	True	False	True	True
season	True	True	False	False	False	True	True	False	False	False
holiday	False	False	True	True	False	False	False	False	False	False
workingday	False	False	True	True	False	False	False	False	False	False
weather	False	False	False	False	True	False	False	True	False	True
temp	True	True	False	False	False	True	True	False	False	True
atemp	True	True	False	False	False	True	True	False	False	True
humidity	False	False	False	False	True	False	False	True	True	True
windspeed	True	False	False	False	False	False	False	True	True	True
Total_Booking	True	False	False	False	True	True	True	True	True	True

From the above method in addition to the alreay eliminated columns we can also eliminate 'Season' column as it has nearly 0 correlation to the target variable- Total_Booking.

7. Task 2: (1.) Feature Engineering :

In this stage the dependent and indepent variables were decided considering the results of all the analysis I had performed till this stage.

X-Train:

	datetime	weather	temp	atemp	humidity	windspeed
0	719163	0	22.14	25.760	77	16.9979
1	719163	0	28.70	33.335	79	19.0012
2	719163	0	5.74	6.060	50	22.0028
3	719163	0	13.94	16.665	29	8.9981
4	719163	0	30.34	33.335	51	19.0012

Similarly X-test, Y_train and Y_test were also defined.

After that Scaling and centering of data using StandardScaler() was also performed.

8. **Task 2: (2.) Grid Search:**

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.model_selection import GridSearchCV

def train_eval(algorithm, grid_params, X_ttrain, X_test, Y_train, Y_test):
    regression_model = GridSearchCV(algorithm, grid_params, cv=5, n_jobs=-1, verbose=1)
    regression_model.fit(X_train, Y_train)
    y_pred = regression_model.predict(X_test)
    print("R2: \t", r2_score(Y_test, y_pred))
    return regression_model
```

Grid Search was performed using the above function.

9. **Task 2: (3.) Regression Analysis:**

Performance of the three Regression Models was conducted. It was judged based on the R2 score since R2 score indicates the predictability of the model.

a) **Linear Regression Model**
The following was the result of Grid Search on Linear Regression Model.

```
Fitting 5 folds for each of 1 candidates, totalling 5 fits
[Parallel(n_jobs=-1)]: Using backend SequentialBackend with 1 concurrent workers.
R2:      0.31542689059214246
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed:    0.8s finished
GridSearchCV(cv=5, error_score='raise-deprecating',
             estimator=LinearRegression(copy_X=True, fit_intercept=True,
                                        n_jobs=None, normalize=False),
             iid='warn', n_jobs=-1, param_grid={}, pre_dispatch='2*n_jobs',
             refit=True, return_train_score=False, scoring=None, verbose=1)
```

b) Decision Tree Regressor Model

The following was the result of Grid Search on Decision Tree Regressor Model.

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits

[Parallel(n_jobs=-1)]: Using backend SequentialBackend with 1 concurrent workers.

R2:      0.35518907836657465

[Parallel(n_jobs=-1)]: Done 500 out of 500 | elapsed:   47.5s finished

GridSearchCV(cv=5, error_score='raise-deprecating',
             estimator=DecisionTreeRegressor(criterion='mse', max_depth=None,
                                             max_features=None,
                                             max_leaf_nodes=None,
                                             min_impurity_decrease=0.0,
                                             min_impurity_split=None,
                                             min_samples_leaf=1,
                                             min_samples_split=2,
                                             min_weight_fraction_leaf=0.0,
                                             presort=False, random_state=None,
                                             splitter='best'),
             iid='warn', n_jobs=-1,
             param_grid={'max_depth': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                         'min_samples_leaf': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=1)
```

c) KNeighborsRegressor Model

The following was the result of Grid Search on KNeighborsRegressor Model.

```
Fitting 5 folds for each of 8 candidates, totalling 40 fits

[Parallel(n_jobs=-1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=-1)]: Done  40 out of  40 | elapsed:  2.7min finished

R2:      0.3802389187607472

GridSearchCV(cv=5, error_score='raise-deprecating',
             estimator=KNeighborsRegressor(algorithm='auto', leaf_size=30,
                                           metric='minkowski',
                                           metric_params=None, n_jobs=None,
                                           n_neighbors=5, p=2,
                                           weights='uniform'),
             iid='warn', n_jobs=-1,
             param_grid={'n_neighbors': [10, 50, 100, 200, 500, 1000, 2000,
                                         5000]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=1)
```

10. **Task 2: (4.) Ensemble Model:**

Similarly , performance of Random Forest Regressor using Grid Search was also performed. I was able to explore Grid Search to it full potential on Random Forest method due to the processing time. If more parameters were given it would have shown better results**.**

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits

[Parallel(n_jobs=-1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=-1)]: Done 500 out of 500 | elapsed: 46.0min finished

R2:      0.43181770350848525

GridSearchCV(cv=5, error_score='raise-deprecating',
             estimator=RandomForestRegressor(bootstrap=True, criterion='mse',
                                             max_depth=None,
                                             max_features='auto',
                                             max_leaf_nodes=None,
                                             min_impurity_decrease=0.0,
                                             min_impurity_split=None,
                                             min_samples_leaf=1,
                                             min_samples_split=2,
                                             min_weight_fraction_leaf=0.0,
                                             n_estimators='warn', n_jobs=None,
                                             oob_score=False, random_state=None,
                                             verbose=0, warm_start=False),
             iid='warn', n_jobs=-1,
             param_grid={'max_depth': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                         'min_samples_leaf': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                         'n_estimators': [100]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=1)
```

Final Inference:

Random Forest Regressor post scaling provided the best R2 score of 0.4318, amongst all the regression models that were executed.