FLIGHT PRICE PREDICTION

MACHINE LEARNING

REGRESSION PROBLEM

Submitted by:

RICHARD PRABHAKAR

ACKNOWLEDGMENT

This includes mentioning of all the references, research papers, data sources, professionals and other resources that helped you and guided you in completion of the project.

Links:-

https://stackoverflow.com/questions/72677752/create-x-train-and-y-train-for-csv-dataset-in-python

https://stackoverflow.com/questions/68794590/how-should-i-predict-target-variable-if-it-is-not-included-in-the-test-data-for

https://www.kaggle.com/datasets/nehalbirla/vehicle-dataset-from-cardekho

Medium.com- blogs on EDA

EDA by Analytics Vidya

Regularized Regression Models Kaggle notebook – by CHOWDHURY SALEH AHMED RONY, KERIMCAN ARSLAN, JAKKI SESHAPANPU

https://www.kaggle.com/code/spscientist/a-simple-tutorial-on-exploratory-data-analysis

Geeksforgeeks.com- Dataframe manipulation

INTRODUCTION

- ## Business Problem Framing

  Anyone who has booked a flight ticket knows how unexpectedly the prices vary. The cheapest available ticket on a given flight gets more and less expensive over time. This usually happens as an attempt to maximize revenue based on -

  1. Time of purchase patterns (making sure last-minute purchases are expensive)

  2. Keeping the flight as full as they want it (raising prices on a flight which is filling up in order to reduce sales and hold back inventory for those expensive last-minute expensive purchases)

  So, I have to work on a project where you collect data of flight fares with other features and work to make a model to predict fares of flights.

- ## Conceptual Background of the Domain Problem

  Predicting flight prices is a common problem in the travel industry, as it can help airlines, online travel agencies, and other companies make more informed pricing decisions, and assist travellers in finding the best deals. There are a number of factors that can influence the price of a flight, including the route, the time of year, the day of the week, and the number of seats available. Additionally, external factors such as fuel prices, currency exchange rates, and economic conditions can also have an impact on flight prices. To predict flight prices, many companies use historical flight data and pricing information to build statistical models, which can then be used to make forecasts about future prices. Machine learning techniques such as linear regression, decision trees, and neural networks are commonly used to build these models. Additionally, a number of companies also make use of web scraping techniques to gather data from airline websites and other sources in order to build a dataset for their models.

- ## Review of Literature

When building a flight price prediction model using data science and machine learning, it is important to review the existing literature on the topic. This literature can provide valuable insights into the factors that influence flight prices, as well as the methods that have been used to predict flight prices in the past. Some key areas to focus on when reviewing the literature include:

Factors that influence flight prices: Researchers have identified a number of factors that can influence flight prices, including route, time of year, day of the week, and number of seats available. Other factors such as fuel prices, currency exchange rates, and economic conditions can also play a role.

- Methods for collecting flight data: In order to build a flight price prediction model, a large dataset of flight prices and other relevant information is needed. Many researchers have used web scraping techniques to collect this data from airline websites and other sources.

- Machine learning methods for flight price prediction: Many researchers have used machine learning techniques to predict flight prices, including linear regression, decision trees, and neural networks. Some studies have also used ensemble methods such as Random Forest and Gradient Boosting to improve the performance of the model.

- Evaluation metrics: Various evaluation metrics have been used to evaluate the performance of flight price prediction models, including mean absolute error, root mean squared error, and mean absolute percentage error.

- Recent Advancement and Trends: The literature on the field is an ever-evolving one, and it's good to keep track of recent advancements and trends, such as the usage of deep learning methods, the integration of external factors such as weather, and the use of more sophisticated features engineering.

- By reviewing the literature on flight price prediction, one can gain a better understanding of the factors that influence flight prices, the methods that have been used to predict flight prices in the past, and the evaluation metrics that have been used to assess the performance of different models. This knowledge can be used to inform the design and development of a flight price prediction model.

- Motivation for the Problem Undertaken

  There are several reasons why predicting flight prices with data science and machine learning is a valuable problem to undertake.

- Increased revenue: By accurately predicting flight prices, airlines and travel agencies can adjust their prices in real-time to better match market demand. This can lead to increased revenue, as they can charge more for high-demand flights, and offer discounts on flights with lower demand.

- Improved customer satisfaction: Accurate flight price predictions can also help travel companies offer their customers more competitive prices. By providing customers with the best deals, companies can improve customer satisfaction and loyalty, which can lead to increased repeat business.

- Optimize Flight Tickets Allocation: Accurately forecasting the demand of a flight also helps airlines optimize their allocation of flight tickets based on the predicted demand.

- Cost Optimization: Predictive modeling can help airlines and travel agencies to optimize their costs as they can predict which flights will be most in demand, and allocate resources accordingly.

- Automation: Automating the process of price prediction with Machine learning models can save a lot of time and effort for the companies instead of doing it manually.

- Overall, predicting flight prices with data science and machine learning can provide significant benefits for both the companies in the travel industry and for customers looking for the best deals on flights

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  Various Statistical modelling used and the concept behind their use-case here:-

  Before model building , we used basic statistical tolls to make analytical decisions to do feature engineering The Tools we used are mean , median , mode, study of quantile's. we used skewness to see if the data was normally distributed, and we used correlation to see if there is similar relationship between the independent features and we used it to see which features are more related to the label. We also used box-cox which is a transformation technique used to remove outliers and then we started using the models such as :-

  1. Linear regression: This is a basic and widely used technique for modeling the relationship between a dependent variable and one or more independent variables. It involves fitting a straight line (or hyperplane in higher dimensions) to the data such that the distance between the data points and the line is minimized. Linear regression can be used for both simple linear regression (one independent variable) and multiple linear regression (more than one independent variable). *Reason*: as we are predicting a continues target variable
  2. Decision tree regression: This is a type of non-linear regression that involves building a decision tree to make

predictions. It is useful for modeling relationships between variables that are not linear.
Reason: As we are deriving the best features of a house we need to predict the price of a house , the decision tree can narrow it down

3. Random forest regression: This is an ensemble method that involves training multiple decision tree models and combining their predictions to make a final prediction. It is often more accurate than a single decision tree, but it is also more computationally expensive
Reason: Similar to Decision tree but multiple trees to get an even better model as with more tree we can cross verify and come up with the best

4. XGBoost (eXtreme Gradient Boosting) is a popular and powerful machine learning algorithm that can be used for both regression and classification tasks. It is an implementation of gradient boosting, which is a type of ensemble method that combines the predictions of multiple weak models to create a strong model.
    a. In the case of regression, XGBoost builds an ensemble of decision trees to make predictions. The algorithm works by training a series of decision tree models in a sequential manner, where each tree is trained to correct the mistakes made by the previous tree. The predictions of all the trees are then combined to make a final prediction.
    b. There are several mathematical, statistical, and analytical techniques that are used in XGBoost regression:
    c. Boosting: As mentioned above, XGBoost is an implementation of gradient boosting, which is a type of boosting algorithm. Boosting algorithms work by training a series of weak models sequentially, where each model is trained to correct the mistakes made by

the previous model. The predictions of all the models are then combined to make a final prediction.

d. Decision trees: XGBoost uses decision trees as the base model for its ensemble. Decision trees are a type of non-linear model that can be used for both regression and classification tasks. They work by dividing the feature space into regions and making predictions based on which region the data belongs to.
e. Gradient descent: XGBoost uses gradient descent to optimize the loss function and find the optimal values for the model parameters. Gradient descent is an iterative optimization algorithm that works by moving in the direction of the negative gradient of the loss function, which helps to minimize the loss.
f. Regularization: XGBoost includes several regularization techniques that can help to prevent overfitting, including L1 and L2 regularization and early stopping. These techniques help to reduce the complexity of the model and improve its generalization performance.
g. Feature importance: XGBoost includes a feature importance feature that can be used to identify which features are most important for making predictions. This can be useful for feature selection and understanding the underlying relationships in the data.

Reason: One of the best models which as explained above uses many forms of statistical models and get the best or the score with the lowest error to make an effective predictor

- Data Sources and their formats

Data was scrapped from Yatra website on domestic flights in India alone

- Data contains 10683 entries each having 11 variables.

- Derivations of the columns in the dataset:_

```
    Airline          10683 non-null  object
    Date_of_Journey  10683 non-null  object
    Source           10683 non-null  object
    Destination      10683 non-null  object
    Route            10682 non-null  object
    Dep_Time         10683 non-null  object
    Arrival_Time     10683 non-null  object
    Duration         10683 non-null  object
    Total_Stops      10682 non-null  object
    Additional_Info  10683 non-null  object
10  Price            10683 non-null  int64
```

- Data Pre-processing Done

The Steps followed in order to clean the data

1. From description we can see that Date_of_Journey is a object data type,
2. Therefore, we have to convert this datatype into timestamp so as to use this column properly for prediction
3.
4. For this we require pandas to_datetime to convert object data type to datetime dtype.
5. Checking for duplicates- found none
6. Checking for null values- found only one , treated with drop function,
7. Splitting the categorical and numerical columns and assigning them to a variable for studying relationships.
8. Used transformation techniques –Transformer to reduce the skewness and also remove any outliers from the data.
9. One can find many ways to handle categorical data. Some of them categorical data are, Nominal data --> data are not in any order --> OneHotEncoder is used in this case.
   Ordinal data --> data are in order --> LabelEncoder is used in this case.

- Data Inputs- Logic- Output Relationships

  When building a flight price prediction model, the inputs to the model would typically include information about the airline, the date of journey, the source and destination, the route, the time of departure, the arrival time, the duration, and the total number of stops. These inputs would be used as features to predict the output, which is the flight's price.

  The relationship between the inputs and the output can be captured in the following steps:

  1. Airline: The airline that operates the flight can have an impact on the flight's price. For example, flights operated by major airlines may be priced higher than flights operated by smaller, regional airlines.

  2. Date of Journey: The date of journey can also influence the flight's price. Prices may be higher during peak travel seasons, such as summer and holidays, and lower during off-peak seasons.

  3. Source and Destination: The source and destination of the flight can also affect the price. Flights between popular destinations or major cities may be priced higher than flights to more remote or less popular destinations.

  4. Route: The route taken by the flight can also influence the price. For example, a non-stop flight between two cities may be priced higher than a flight with one or more stops.

5. Time of Departure and Arrival: The time of departure and arrival can also have an impact on the price. For example, flights departing or arriving during peak hours may be priced higher than flights departing or arriving during off-peak hours.

6. Duration: The duration of the flight can also have an impact on the price, longer flights tend to be more expensive than shorter flights.

7. Total stops: The total number of stops made during the flight can also affect the price. A flight with multiple stops may be priced lower than a non-stop flight.

8. Other external factors like fuel prices, currency exchange rate and economic conditions can also play a role in the flight's price.

All these factors are taken into account by the model, and the relationships between the inputs and the output are captured in the model's logic. The model would use the input features to make predictions about the flight's price, based on the historical data it has been trained on.

- Hardware and Software Requirements and Tools Used

Listing

```
import pandas as pd ## pandas is used to manupulate the dataframe

import numpy as np ## numpy is used to do scientific calculations

import matplotlib.pyplot as plt ## matplotlib used for visualization or graphs

import seaborn as sns ## seaborn used for visualization or graphs

import missingno as msno ## used to visualize missing values

import warnings ## used to remove warnings

warnings.filterwarnings('ignore')

%matplotlib inline


from sklearn.model_selection import train_test_split,cross_validate, GridSearchCV – ## used to split training data and test , in this case we didn’t use as we have already separate files for training and testing

from sklearn.preprocessing import StandardScaler,OrdinalEncoder# to convert or encode the categorical or string values into numbers

from sklearn.metrics import mean_squared_error #it is a metric used to check the error we get with each model , lower the better

from sklearn.linear_model import LinearRegression,Lasso,Ridge,BayesianRidge ## linear regression model used to predict and train data

from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor # Ensemle techniques used to work on model to see which is better at prediction and lower error

from xgboost import XGBRegressor # another machine learning model but boosting techniques

from lightgbm import LGBMRegressor # another Boosting technique
```

```
import math #to do mathematics based functions on the data be it for visualization or for any cleaning as well

from IPython.display import Image # to save and show image

import warnings # to remove the warnings to show clean outputs

warnings.filterwarnings("ignore")

sns.set(rc={"figure.figsize": (20, 15)})

sns.set_style("whitegrid")
```

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

Also explained in statistical modelling and analytical in previous Analytical Problem Framing Chapter, we use the following methods:-

1. Linear regression: This is a basic and widely used technique for modeling the relationship between a dependent variable and one or more independent variables. It involves fitting a straight line (or hyperplane in higher dimensions) to the data such that the distance between the data points and the line is minimized. Linear regression can be used for both simple

linear regression (one independent variable) and multiple linear regression (more than one independent variable).
Reason: as we are predicting a continues target variable

2. Decision tree regression: This is a type of non-linear regression that involves building a decision tree to make predictions. It is useful for modeling relationships between variables that are not linear.
   Reason: As we are deriving the best features of a house we need to predict the price of a house , the decision tree can narrow it down
3. Random forest regression: This is an ensemble method that involves training multiple decision tree models and combining their predictions to make a final prediction. It is often more accurate than a single decision tree, but it is also more computationally expensive
   Reason: Similar to Decision tree but multiple trees to get an even better model as with more tree we can cross verify and come up with the best

4. XGBoost (eXtreme Gradient Boosting) is a popular and powerful machine learning algorithm that can be used for both regression and classification tasks. It is an implementation of gradient boosting, which is a type of ensemble method that combines the predictions of multiple weak models to create a strong model.
   a. In the case of regression, XGBoost builds an ensemble of decision trees to make predictions. The algorithm works by training a series of decision tree models in a sequential manner, where each tree is trained to correct the mistakes made by the previous tree. The predictions of all the trees are then combined to make a final prediction.
   b. There are several mathematical, statistical, and analytical techniques that are used in XGBoost regression:

c. Boosting: As mentioned above, XGBoost is an implementation of gradient boosting, which is a type of boosting algorithm. Boosting algorithms work by training a series of weak models sequentially, where each model is trained to correct the mistakes made by the previous model. The predictions of all the models are then combined to make a final prediction.
d. Decision trees: XGBoost uses decision trees as the base model for its ensemble. Decision trees are a type of non-linear model that can be used for both regression and classification tasks. They work by dividing the feature space into regions and making predictions based on which region the data belongs to.
e. Gradient descent: XGBoost uses gradient descent to optimize the loss function and find the optimal values for the model parameters. Gradient descent is an iterative optimization algorithm that works by moving in the direction of the negative gradient of the loss function, which helps to minimize the loss.
f. Regularization: XGBoost includes several regularization techniques that can help to prevent overfitting, including L1 and L2 regularization and early stopping. These techniques help to reduce the complexity of the model and improve its generalization performance.
g. Feature importance: XGBoost includes a feature importance feature that can be used to identify which features are most important for making predictions. This can be useful for feature selection and understanding the underlying relationships in the data.

Reason: One of the best models which as explained above uses many forms of statistical models and get the best or the score with the lowest error to make an effective predictor

- Testing of Identified Approaches (Algorithms)

Linear Regression alone and with Tuning techniques,Ridge Regression,Random forest Regressor with and withput hyper parameter tuning techniques,Decision tree Regressor with and without tuning,Xgboost Regressor with and without tuning

- Run and Evaluate selected models
  Snapshot of results with all models :-

Creating the Model - Choosing the Best Model

Linear Regression Model

In [44]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from sklearn.model_selection import train_test_split
```

In [45]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [46]:
```
scores=[]
for i in range(0,1000):
    X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.25,random_state = i)
    lr.fit(X_train,y_train)
    pred_train = lr.predict(X_train)
    pred_test = lr.predict(X_test)
    print(f"At random state {i},the training accuracy is :-{r2_score(y_train,pred_train)}")
    print(f"At random state {i},the testing accuracy is :-{r2_score(y_test,pred_test)}")
    print('\n')
    scores.append(r2_score(y_test,pred_test))
```

```
At random state 0,the training accuracy is :-0.6345097401007789
At random state 0,the testing accuracy is :-0.5897613068908487


At random state 1,the training accuracy is :-0.6284963534837915
At random state 1,the testing accuracy is :-0.605686642811051


At random state 2,the training accuracy is :-0.6298854957958571
At random state 2,the testing accuracy is :-0.6024142307021293


At random state 3,the training accuracy is :-0.6270949586090333
At random state 3,the testing accuracy is :-0.6107242760087814


At random state 4,the training accuracy is :-0.6237600500172336
At random state 4,the testing accuracy is :-0.6214124683098589
```

```
At cross fold2 the cv score is 0.6183722734703163 and accuracy score for training is -0.6476730474528358and the accuracy for te
sting is 0.679551129054491


At cross fold3 the cv score is 0.6190349040445325 and accuracy score for training is -0.6476730474528358and the accuracy for te
sting is 0.679551129054491


At cross fold4 the cv score is 0.6213062350335223 and accuracy score for training is -0.6476730474528358and the accuracy for te
sting is 0.679551129054491


At cross fold5 the cv score is 0.6199027318916419 and accuracy score for training is -0.6476730474528358and the accuracy for te
sting is 0.679551129054491


At cross fold6 the cv score is 0.6205586532825998 and accuracy score for training is -0.6476730474528358and the accuracy for te
sting is 0.679551129054491


At cross fold7 the cv score is 0.6195867852859553 and accuracy score for training is -0.6476730474528358and the accuracy for te
sting is 0.679551129054491


At cross fold8 the cv score is 0.6199893507866636 and accuracy score for training is -0.6476730474528358and the accuracy for te
sting is 0.679551129054491


At cross fold9 the cv score is 0.6194440237606135 and accuracy score for training is -0.6476730474528358and the accuracy for te
sting is 0.679551129054491
```

Regularization of the Linear Model

In [56]:
```
from sklearn.model_selection import GridSearchCV #to select the pest parameters for hyperparameter tuning
from sklearn.model_selection import cross_val_score #to check the difference from the earlier score without hyper parameter tunir
```

In [57]:
```
from sklearn.linear_model import Lasso

parameters ={'alpha' : [.0001, .001, .01, .1, 1, 10],
             'random_state' : list(range(0,15))}
ls = Lasso()
clf = GridSearchCV(ls,parameters)
clf.fit(X_train, y_train)

print(clf.best_params_)
```

```
{'alpha': 0.1, 'random_state': 0}
```

Final model training for Linear Regression

In [58]:
```
ls = Lasso(alpha= 0.1, random_state= 0)
ls.fit(X_train,y_train)
ls_score_training = ls.score(X_train,y_train)
pred_ls = ls.predict(X_test)
ls_score_training*100
```

Out[58]:
```
59.929973516550675
```

Checking MSE,RMSE score

In [59]:
```
from sklearn import metrics

print('MAE:', metrics.mean_absolute_error(y_test, pred_test))
print('MSE:', metrics.mean_squared_error(y_test, pred_test))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, pred_test)))
```

```
MAE: 1971.9149809559103
MSE: 7832335.979313649
RMSE: 2798.6310902499545
```

Decision Tree Regressor

In [60]:
```
from sklearn.tree import DecisionTreeRegressor

dt=DecisionTreeRegressor()
dt.fit(X_train,y_train)
dt.score(X_train,y_train)
pred_test =dt.predict(X_test)
dfs = r2_score(y_test,pred_test)
print('R2 Score :',dfs*100)

dfscore = cross_val_score(dt,X,y,cv=9)
dfc =dfscore.mean()
print('Cross Val Score :',dfc*100)
```

```
R2 Score : 67.28187183479127
Cross Val Score : 71.84741717126788
```

In [61]:
```
from sklearn import metrics

print('MAE:', metrics.mean_absolute_error(y_test, pred_test))
print('MSE:', metrics.mean_squared_error(y_test, pred_test))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, pred_test)))
```

```
MAE: 1401.5498252839136
MSE: 7996887.979291256
RMSE: 2827.8769384984303
```

SVR

In [67]:
```
from sklearn.svm import SVR
svr= SVR()

svr.fit(X_train,y_train)
svr.score(X_train,y_train)
pred_decision =svr.predict(X_test)

svrs = r2_score(y_test,pred_decision)
print('R2 Score :',svrs*100)

svrscore = cross_val_score(svr,X,y,cv=2)
svrc =svrscore.mean()
print('Cross Val Score :',svrc*100)
```

```
R2 Score : 0.15235660771393267
Cross Val Score : -0.6282878815062787
```

In [68]:
```
#Checking MAE MSE and RMSE scores
from sklearn import metrics

print('MAE:', metrics.mean_absolute_error(y_test, pred_decision))
print('MSE:', metrics.mean_squared_error(y_test, pred_decision))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, pred_decision)))
```

```
MAE: 3591.438223880594
MSE: 24404526.297240835
RMSE: 4940.093753891806
```

K- Nearest Neighbors

In [62]:
```
from sklearn.neighbors import KNeighborsRegressor

knn=KNeighborsRegressor()
knn.fit(X_train,y_train)
knn.score(X_train,y_train)
pred_test =knn.predict(X_test)
knns = r2_score(y_test,pred_test)
print('R2 Score :',knns*100)

knnscore = cross_val_score(knn,X,y,cv=9)
knnc =knnscore.mean()
print('Cross Val Score :',knnc*100)
```

```
R2 Score : 52.83591953559734
Cross Val Score : 59.03021303135111
```

In [63]:
```
from sklearn import metrics

print('MAE:', metrics.mean_absolute_error(y_test, pred_test))
print('MSE:', metrics.mean_squared_error(y_test, pred_test))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, pred_test)))
```

```
MAE: 1933.4228378884313
MSE: 11527733.683773868
RMSE: 3395.2516377691163
```

Xgboost Regressor

In [69]:
```
import xgboost as xgb
xgb = xgb.XGBRegressor()

xgb.fit(X_train,y_train)
xgb.score(X_train,y_train)
pred_decision =xgb.predict(X_test)

xgbs = r2_score(y_test,pred_decision)
print('R2 Score :',xgbs*100)

xgbscore = cross_val_score(xgb,X,y,cv=9)
xgbc =xgbscore.mean()
print('Cross Val Score :',xgbc*100)
```

```
R2 Score : 85.73561497458645
Cross Val Score : 84.67915379336922
```

In [70]:
```
#Checking MAE MSE and RMSE scores
from sklearn import metrics

print('MAE:', metrics.mean_absolute_error(y_test, pred_decision))
print('MSE:', metrics.mean_squared_error(y_test, pred_decision))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, pred_decision)))
```

Hyperparameter Tuning

- Choose following method for hyperparameter tuning
  1. **RandomizedSearchCV** --> Fast
  2. **GridSearchCV**
- Assign hyperparameters in form of dictionery
- Fit the model
- Check best paramters and best score

In [62]:
```
from sklearn.model_selection import RandomizedSearchCV
```

In [63]:
```
#Randomized Search CV

# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10, 15, 100]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 5, 10]
```

In [64]:
```
# Create the random grid

random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf}
```

In [65]:
```
# Random search of parameters, using 5 fold cross validation,
# search across 100 different combinations
rf_random = RandomizedSearchCV(estimator = reg_rf, param_distributions = random_grid,scoring='neg_mean_squared_error', n_iter = 1
```

In [66]:
```
rf_random.fit(X_train,y_train)
```

```
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:   12.9s remaining:    0.0s

[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10, total=  11.5s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10, total=  11.8s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10, total=  12.4s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10, total=  10.9s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15, total=  18.6s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15, total=  17.9s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15, total=  18.5s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15, total=  16.6s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15, total=  16.6s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto, max_depth=15
```

In [67]:
```
rf_random.best_params_
```

Out[67]:
```
{'n_estimators': 700,
 'min_samples_split': 15,
 'min_samples_leaf': 1,
 'max_features': 'auto',
 'max_depth': 20}
```

```
In [72]: #Checking MAE MSE and RMSE scores
         from sklearn import metrics

         print('MAE:', metrics.mean_absolute_error(y_test, pred_decision))
         print('MSE:', metrics.mean_squared_error(y_test, pred_decision))
         print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, pred_decision)))

         MAE: 1165.606162629916
         MSE: 4062650.6911608884
         RMSE: 2015.6018186042818
```

Save the model to reuse it again

```
In [62]: import pickle
         # open a file, where you ant to store the data
         file = open('flight_rf.pkl', 'wb')

         # dump information to that file
         pickle.dump(reg_rf, file)

In [63]: model = open('flight_price_rf.pkl','rb')
         forest = pickle.load(model)

In [64]: y_prediction = forest.predict(X_test)

In [65]: metrics.r2_score(y_test, pred_decision)

Out[65]: 0.8117443234361064
```

Best model is XGB Regressor with 85% score

- Key Metrics for success in solving problem under consideration

R2 Score

The R2 score, also known as the coefficient of determination, is a measure of the goodness of fit of a machine learning model. It is used to evaluate the performance of a regression model, and it can range from 0 to 1, with a higher value indicating a better fit.

The R2 score is calculated by taking the sum of the squares of the differences between the predicted values and the actual values, and dividing it by the sum of the squares of the differences between the actual values and the mean of the actual values. This results in a value between 0 and 1, with 0 indicating that the model is not a good fit and 1 indicating a perfect fit.

For example, if a model has an R2 score of 0.8, this means that the model explains 80% of the variance in the data. On the other hand,

if a model has an R2 score of 0.2, this means that the model only explains 20% of the variance in the data, which may indicate that the model is not performing well.

In general, the R2 score is a useful metric for evaluating the performance of a machine learning model, especially when the goal is to predict a continuous target variable. However, it is important to keep in mind that the R2 score can be affected by the number of variables in the model, and it may not always be the most appropriate metric to use, depending on the specific characteristics of the data and the goals of the analysis.

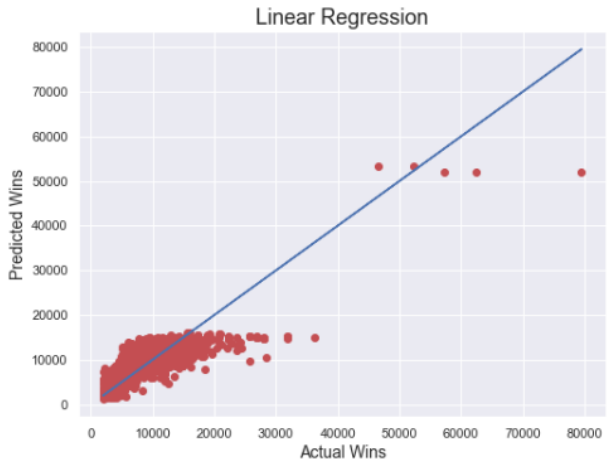
Result : we see that we are able to achieve 85% accuracy with Xgboost Regressor model

- Visualizations

Linear regression plots

Plotting the linear Regression graph with actual and predicted values comparison

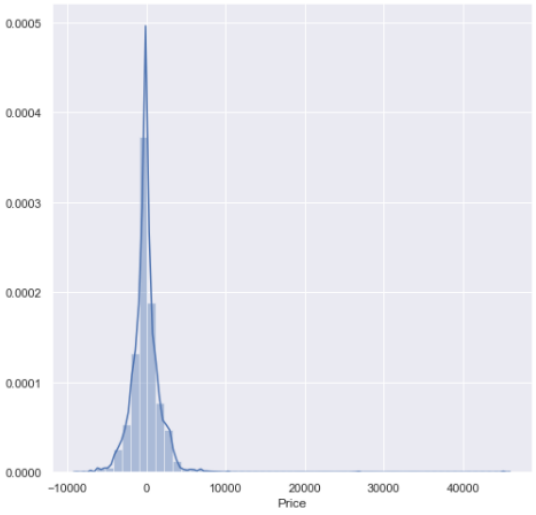
```
5]: import matplotlib.pyplot as plt
plt.figure(figsize=(8,6))
plt.scatter(x=y_test, y=pred_test,color='r')
plt.plot(y_test,y_test,color='b')
plt.xlabel('Actual Wins',fontsize=14)
plt.ylabel('Predicted Wins',fontsize=14)
plt.title('Linear Regression',fontsize=18)
plt.show()
```



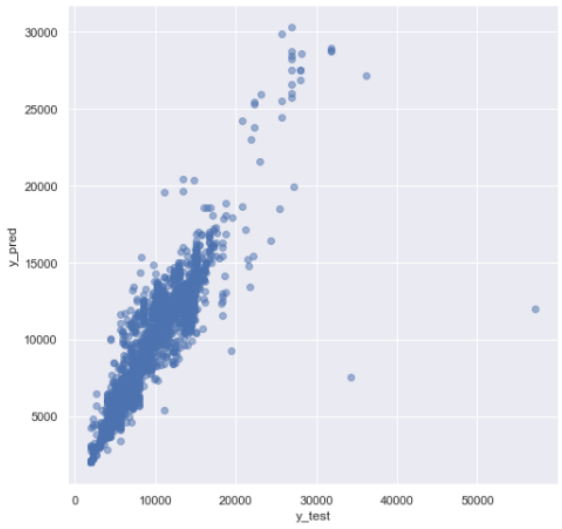
NOT DOING GOOD AT ALL , USING RANDOM FOREST NOW

In [69]:
```
prediction = rf_random.predict(X_test)
```

In [70]:
```
plt.figure(figsize = (8,8))
sns.distplot(y_test-prediction)
plt.show()
```



In [71]:
```
plt.figure(figsize = (8,8))
plt.scatter(y_test, prediction, alpha = 0.5)
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()
```



WE SEE MUCH BETTER DISTRIBUTION IN RANDOM FOREST MODEL

- Interpretation of the Results

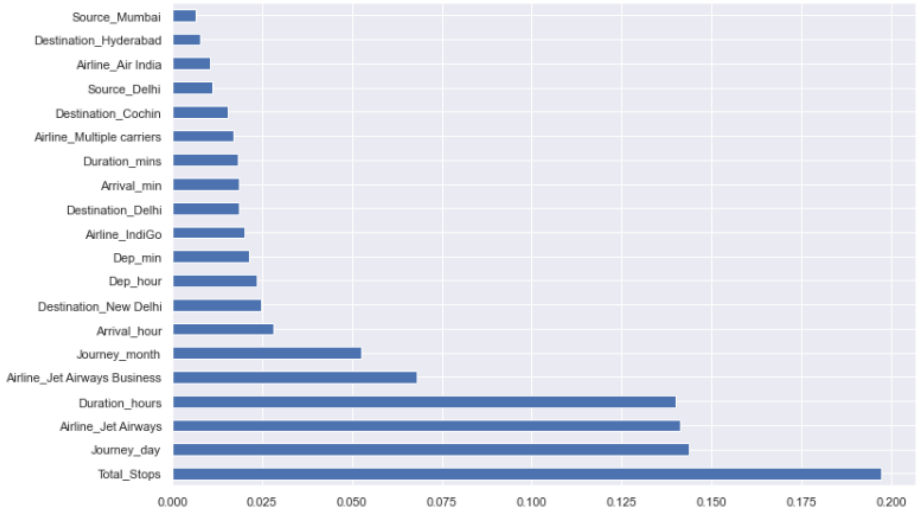
From the data set we see that as we have specifically replaced the null values in many of the columns with the right replacements and we have treated the dataset for outliers therefore , we were able to get a good accuracy score , we can actually used other metrics like mean squared error as well and get 0.11 to .14% which is very good , but the accuracy score being 85% + shows that the model is very good and can be used for predictions for flight prices , we saw that few features were influencing the price way more than others

```
In [42]: print(selection.feature_importances_)
```

```
[1.97032451e-01 1.43794786e-01 5.25303744e-02 2.36184704e-02
 2.14368332e-02 2.82449910e-02 1.85163591e-02 1.40059984e-01
 1.83510247e-02 1.06206723e-02 1.94937041e-03 2.00137844e-02
 1.41333272e-01 6.79654437e-02 1.69092382e-02 8.16011248e-04
 3.03355187e-03 1.18664472e-04 4.82308034e-03 8.23423411e-05
 4.91490965e-04 1.11468736e-02 3.33146071e-03 6.45706010e-03
 1.56090622e-02 1.86387516e-02 7.68761015e-03 5.02396437e-04
 2.48845886e-02]
```

```
In [43]: #plot graph of feature importances for better visualization

plt.figure(figsize = (12,8))
feat_importances = pd.Series(selection.feature_importances_, index=X.columns)
feat_importances.nlargest(20).plot(kind='barh')
plt.show()
```



Achieving this score is a good sign that data science is really going to make a positive impact in the business decisions taken by management on the flight price project

CONCLUSION

- Key Findings and Conclusions of the Study

Describe

From the EDA and graphs we saw:

1. Airline: The airline that operates the flight can have an impact on the flight's price. For example, flights operated by major airlines may be priced higher than flights operated by smaller, regional airlines.

2. Date of Journey: The date of journey can also influence the flight's price. Prices may be higher during peak travel seasons, such as summer and holidays, and lower during off-peak seasons.

3. Source and Destination: The source and destination of the flight can also affect the price. Flights between popular destinations or major cities may be priced higher than flights to more remote or less popular destinations.

4. Route: The route taken by the flight can also influence the price. For example, a non-stop flight between two cities may be priced higher than a flight with one or more stops.

5. Time of Departure and Arrival: The time of departure and arrival can also have an impact on the price. For example, flights departing or arriving during peak hours may

be priced higher than flights departing or arriving during off-peak hours.

6. Duration: The duration of the flight can also have an impact on the price, longer flights tend to be more expensive than shorter flights.

7. Total stops: The total number of stops made during the flight can also affect the price. A flight with multiple stops may be priced lower than a non-stop flight.

8. Other external factors like fuel prices, currency exchange rate and economic conditions can also play a role in the flight's price.

All these factors are taken into account by the model, and the relationships between the inputs and the output are captured in the model's logic. The model would use the input features to make predictions about the flight's price, based on the historical data it has been trained on.

- Learning Outcomes of the Study in respect of Data Science

As mentioned in the previous part we have many features which have the most influence on the Target and having very impactful features it was really difficult to see prior to model building to come up with a good score or a good model , but since in Machine learning the computation is not statistical alone and the model actually learns the data as a whole , we are able to see such good scores. We see that in most situations we can go with Xgboost regressor as the model is not having overfitting or underfitting , it’s a strong model as well. If we are to use a substitute we can go for

Random Forest regression as well as we saw it has a close score to Xgboost but the training was lesser, , But overall has a better and faster processing time so XGBoost regressor is the clear winner

- Limitations of this work and Scope for Future Work

Flight price prediction models using data science and machine learning can be powerful tools for predicting future flight prices, but they do have some limitations. Some of the limitations include:

- Data limitations: The accuracy of the prediction model is highly dependent on the quality and quantity of the data used to train the model. If the data is incomplete or inaccurate, the model's predictions may also be inaccurate. Additionally, the model may not generalize well to new, unseen data if the training data is not diverse enough.

- Limited to Historical Data: The model relies on historical data and may not account for changes in the market, such as new competitors entering the market, or changes in government regulations.

- Limited to the factors included in the model: There may be other factors that influence flight prices that are not included in the model, such as weather conditions or political events.

- Overfitting: Overfitting may occur when the model is too complex and captures noise in the data, which can lead to poor generalization to new data.

- No account of human factors: The model cannot account for human factors such as the decision of the airline to lower the

price to fill the seats or sudden price hike due to unexpected high demand.

Despite these limitations, flight price prediction models using data science and machine learning can still be powerful tools for making more informed pricing decisions, and for helping customers find the best deals on flights.

In future work, to overcome these limitations, one can consider incorporating external data sources such as weather, currency exchange rate, fuel prices, and other relevant information to improve the model's performance. Another approach would be to incorporate more sophisticated techniques such as deep learning and reinforcement learning to improve the model's ability to make predictions. Additionally, one can also incorporate more sophisticated feature engineering techniques to improve the performance of the model.