Flight Project:

|  |  |  |
| --- | --- | --- |
| **Price Project** | | **Satisfaction Project** |
| **Raw File** | | |
| Flight\_Price.csv | | Passenger\_Satisfaction.csv |
| **EDA and Data Preprocessing File** | | |
| D:\GUVI\visual\_studio\Fligh Data\flight\_project\price\_data\_analysis.ipynb | |  |
| **Data Files used for the ML Models** | | |
| Flight\_price\_cleaned\_data\_2 | | amputed\_delay\_distance.csv |
|  | Converted duration in min from object to integer |  |
|  | Extracted total number of stops (int) from the route (counting “?”)  Function:  def split\_dur(x):      pattern1 = "\d+h"      pattern2 = "\d+m"      h = re.findall(pattern1, x)      m = re.findall(pattern2, x)      try:          h = int(h[0].replace('h', ''))      except:          h=0      try:          m = int(m[0].replace('m', ''))      except:          m=0      return (h\*60)+m  application:  data['dur\_min'] = data.Duration.apply(lambda x: split\_dur(x)) |  |
|  | Extracted day of week from journey date (int)  Day of month is not normally distributed and days have limited dates in each month, so it is better to choose day of week over day of month. |  |
|  | Converted departure time into integer using:  dep\_time= dep\_time.apply(lambda x: int(x.strftime("%H%M%S"))) |  |
|  | Additional Info column: replaced No Info to No info . Case matching data.Additional\_Info.replace('No info', 'No Info', inplace=True) |  |
|  | Imputing outliers in price – airline wise  (this should be price and route wise imputation) |  |
|  | Imputing outliers in Duration by Routes |  |
|  | Dropped Route, Journey Date and Arrival Time |  |
|  | Saved data to csv file: Flight\_price\_cleaned\_data\_2 |  |
| Models | |  |
| models = [      (          'LinearRegression',          {'n\_jobs': None},          LinearRegression(),          (x\_train, y\_train),          (x\_test, y\_test)      ),      (          'Decision Tree',          {'criterion': 'absolute\_error',          'max\_depth': 7,          'min\_samples\_leaf': 70,          'min\_samples\_split': 220},          DecisionTreeRegressor(),          (x\_train, y\_train),          (x\_test, y\_test)      ),      (          'Decision Tree CV',          {'criterion': 'friedman\_mse',          'max\_depth': 25,          'max\_features': 32,          'min\_samples\_leaf': 75,          'min\_samples\_split': 210},          DecisionTreeRegressor(),          (x\_train, y\_train),          (x\_test, y\_test)      ),        (          'Gradient Boosting',          {'learning\_rate': 0.1,          'max\_depth': 3,          'n\_estimators': 50,          },          GradientBoostingRegressor(),          (x\_train, y\_train),          (x\_test, y\_test)      ),        (          'Gradient Boosting CV',          {'learning\_rate': 0.3,          'max\_depth': 6,          'n\_estimators': 65,          },          GradientBoostingRegressor(),          (x\_train, y\_train),          (x\_test, y\_test)      ),        (          'Random Forest',          {'max\_depth': None,          'min\_samples\_leaf':1,          'min\_samples\_split':2,          'n\_estimators': 100,          },          RandomForestRegressor(),          (x\_train, y\_train),          (x\_test, y\_test)      ),        (          'Random Forest CV',          {'max\_depth': 5,          'min\_samples\_leaf':1,          'min\_samples\_split':2,          'n\_estimators': 125,          },          RandomForestRegressor(),          (x\_train, y\_train),          (x\_test, y\_test)      )  ] | |  |
| Registering Model to MLFlow | |  |
| mlflow.set\_experiment('Flight\_Price\_Experiment\_0')  mlflow.set\_tracking\_uri('http://127.0.0.1:5000')  for i, element in enumerate(models):      model\_name = element[0]      params = element[1]      model = element[2]      score = report[i]        with mlflow.start\_run(run\_name = model\_name):          mlflow.log\_params(params)          mlflow.log\_metrics({'MSE': score[1],                              'RMSE':score[2],                              'MAE':score[3],                              'R2': score[4]})          mlflow.sklearn.log\_model(model, 'model') | |  |