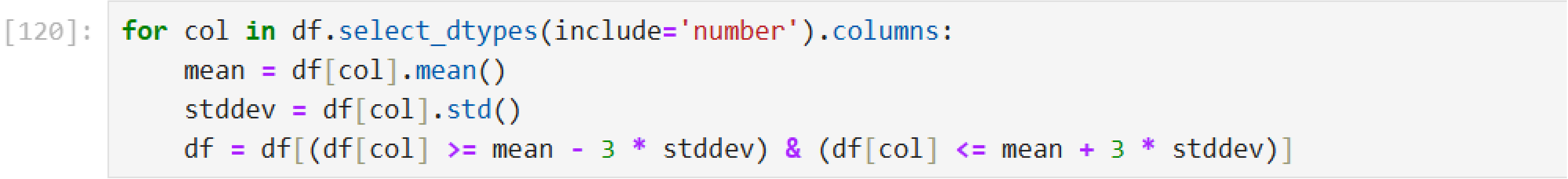
Cellula Technologies – Task(5)

- Internship 1 – Group 5

- Malak Ahmed Saber
- Marawan Abbas Mohamed

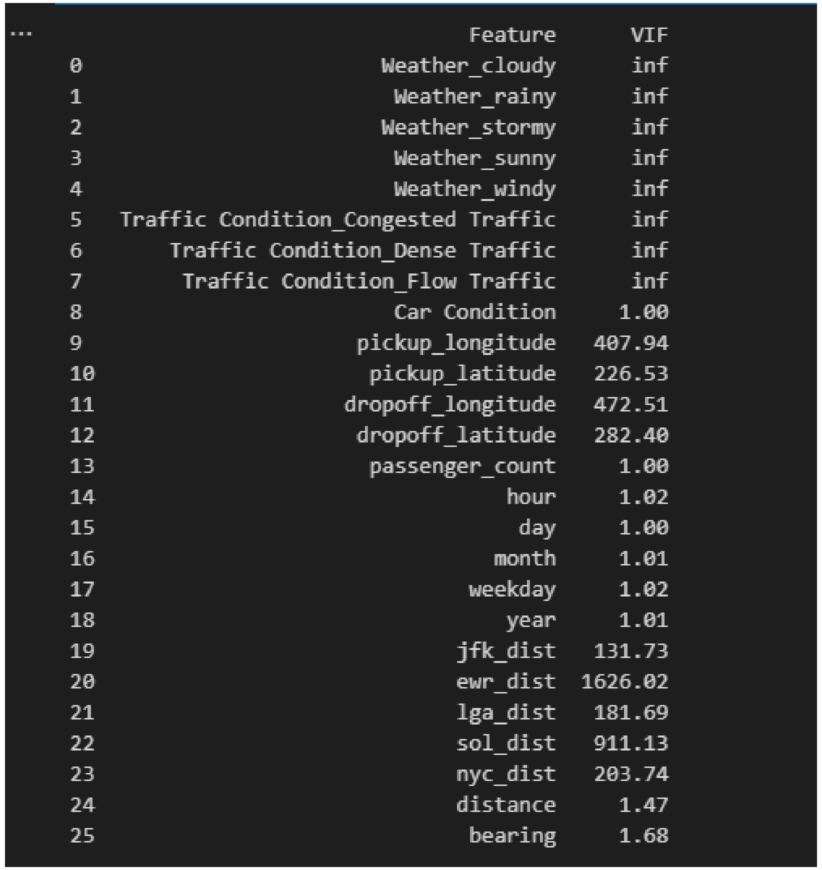
Data Overview

- 500000 row x 26 features about Drivers names, fare amount, weather condition, traffic condition and more
- 9 columns with 5 common null values, filled with their median
- 0 Duplicates
- Outliers :
    - Detected with boxplots
    - Removed with std dev method / used also Z_score as another method



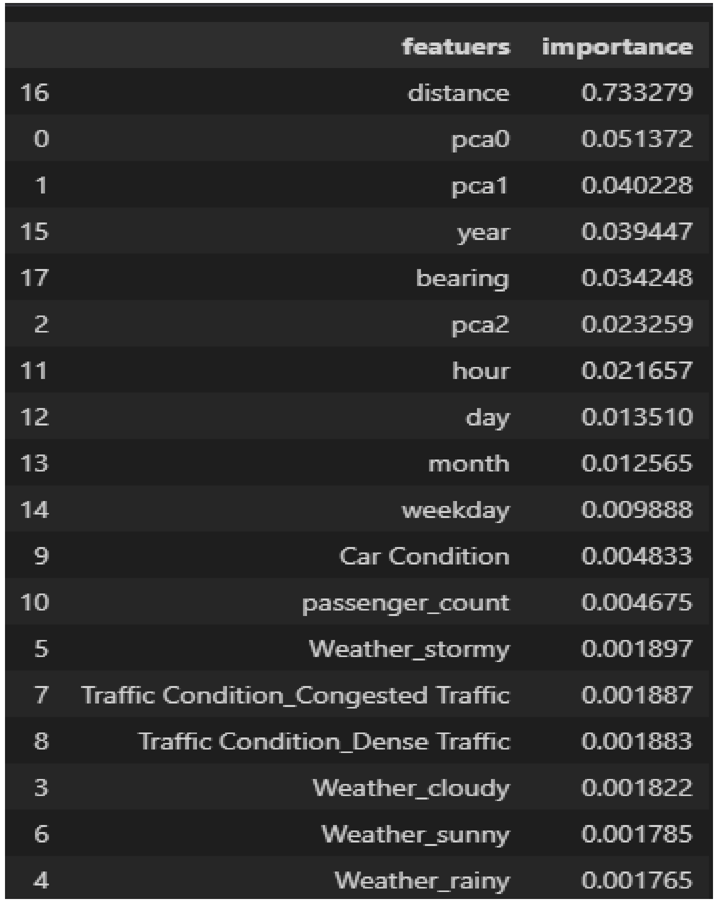
```
[120]: for col in df.select_dtypes(include='number').columns:
           mean = df[col].mean()
           stddev = df[col].std()
           df = df[(df[col] >= mean - 3 * stddev) & (df[col] <= mean + 3 * stddev)]
```

Feature selection



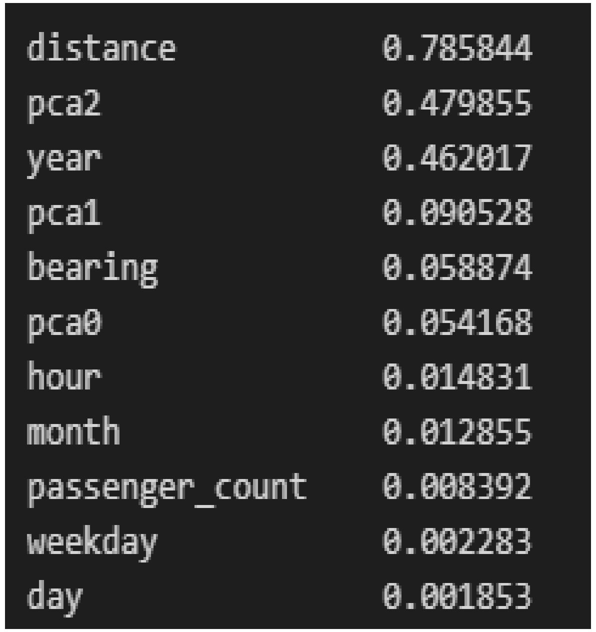
```
...
                                  Feature      VIF
0                           Weather_cloudy      inf
1                            Weather_rainy      inf
2                           Weather_stormy      inf
3                            Weather_sunny      inf
4                            Weather_windy      inf
5      Traffic Condition_Congested Traffic      inf
6          Traffic Condition_Dense Traffic      inf
7           Traffic Condition_Flow Traffic      inf
8                            Car Condition     1.00
9                         pickup_longitude   407.94
10                         pickup_latitude   226.53
11                       dropoff_longitude   472.51
12                        dropoff_latitude   282.40
13                         passenger_count     1.00
14                                    hour     1.02
15                                     day     1.00
16                                   month     1.01
17                                 weekday     1.02
18                                    year     1.01
19                                jfk_dist   131.73
20                                ewr_dist  1626.02
21                                lga_dist   181.69
22                                sol_dist   911.13
23                                nyc_dist   203.74
24                                distance     1.47
25                                 bearing     1.68
```

Making sure features aren't strongly related by Checking multicollinearity



	featuers	importance
16	distance	0.733279
0	pca0	0.051372
1	pca1	0.040228
15	year	0.039447
17	bearing	0.034248
2	pca2	0.023259
11	hour	0.021657
12	day	0.013510
13	month	0.012565
14	weekday	0.009888
9	Car Condition	0.004833
10	passenger_count	0.004675
5	Weather_stormy	0.001897
7	Traffic Condition_Congested Traffic	0.001887
8	Traffic Condition_Dense Traffic	0.001883
3	Weather_cloudy	0.001822
6	Weather_sunny	0.001785
4	Weather_rainy	0.001765

Used RandomForestRegressor to find the most important features.



```
distance           0.785844
pca2               0.479855
year               0.462017
pca1               0.090528
bearing            0.058874
pca0               0.054168
hour               0.014831
month              0.012855
passenger_count    0.008392
weekday            0.002283
day                0.001853
```

Checked mutual information as an extra measure for feature importance and take largest 10.

Feature Engineering

Categorical Columns:

- Dropped : 'User ID', 'User Name', 'key', 'pickup_datetime’
- Label Encoding :’Car Condition', ‘Weather, ‘Traffic Condition’
- Frequency Encoding : ‘Driver Name’ then dropped the original column

Numerical Columns:

- PCA Applied on : pickup_longitude', 'pickup_latitude','dropoff_longitude', 'dropoff_latitude', 'jfk_dist', 'ewr_dist', 'lga_dist','sol_dist', 'nyc_dist
- PCA applied due to high multicollinearity

Train Test Split

Target
‘fare amount’

Train : Test
80% : 20%

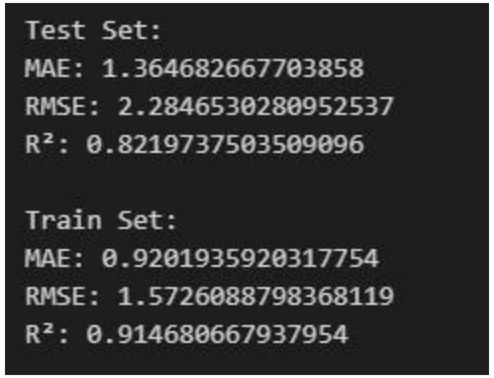
Random State
42

Modelling

- Models Used :
  - Random Forest Regressor
  - Linear Regression
  - KNN
  - XGBOOST
  - Ridge Regression
  - Decision Tree

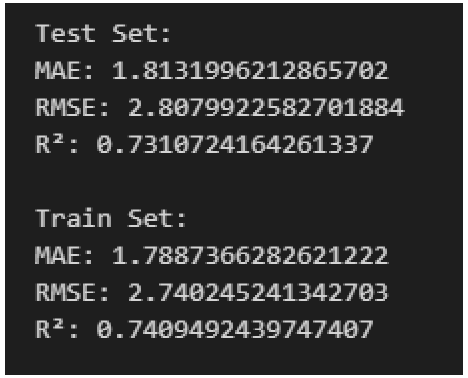
Models Evaluation

Random Forest



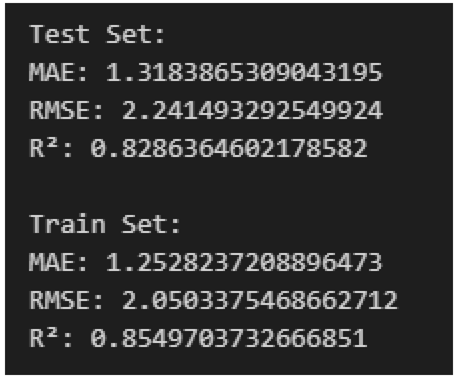
```
Test Set:
MAE: 1.364682667703858
RMSE: 2.2846530280952537
R²: 0.8219737503509096

Train Set:
MAE: 0.9201935920317754
RMSE: 1.5726088798368119
R²: 0.914680667937954
```

Linear Regression



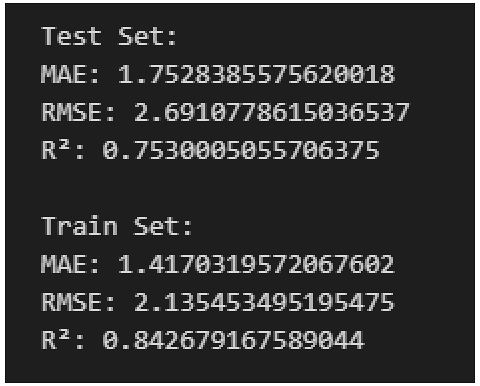
```
Test Set:
MAE: 1.8131996212865702
RMSE: 2.8079922582701884
R²: 0.7310724164261337

Train Set:
MAE: 1.7887366282621222
RMSE: 2.740245241342703
R²: 0.7409492439747407
```

XGBOOST



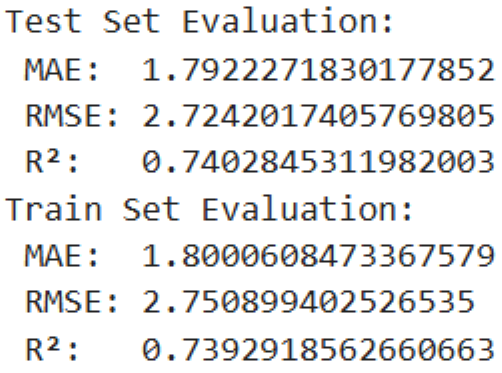
```
Test Set:
MAE: 1.3183865309043195
RMSE: 2.241493292549924
R²: 0.8286364602178582

Train Set:
MAE: 1.2528237208896473
RMSE: 2.0503375468662712
R²: 0.8549703732666851
```

KNN



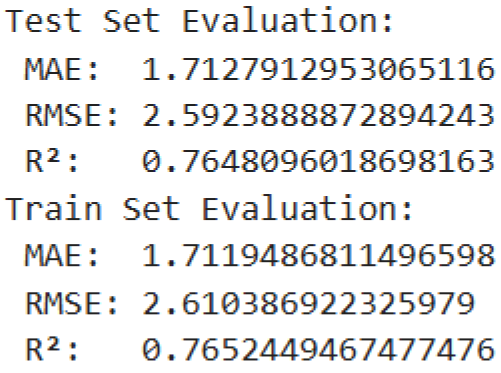
```
Test Set:
MAE: 1.7528385575620018
RMSE: 2.6910778615036537
R²: 0.7530005055706375

Train Set:
MAE: 1.4170319572067602
RMSE: 2.135453495195475
R²: 0.842679167589044
```

Ridge Regression



```
Test Set Evaluation:
 MAE:  1.7922271830177852
 RMSE: 2.7242017405769805
 R²:   0.7402845311982003
Train Set Evaluation:
 MAE:  1.8000608473367579
 RMSE: 2.750899402526535
 R²:   0.7392918562660663
```

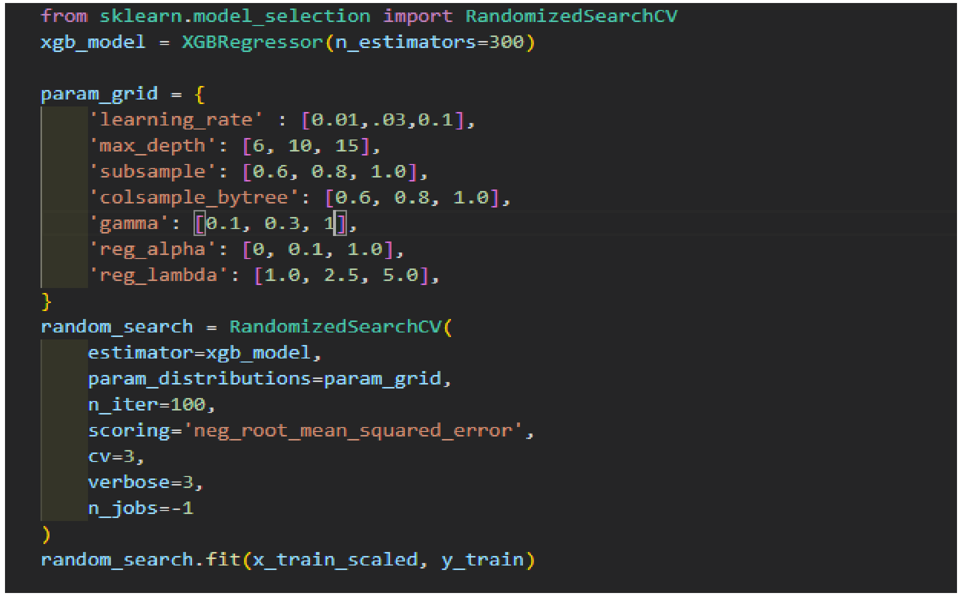
Decision Tree



```
Test Set Evaluation:
 MAE:  1.7127912953065116
 RMSE: 2.5923888872894243
 R²:   0.7648096018698163
Train Set Evaluation:
 MAE:  1.7119486811496598
 RMSE: 2.610386922325979
 R²:   0.7652449467477476
```

Hyperparameter Tuning

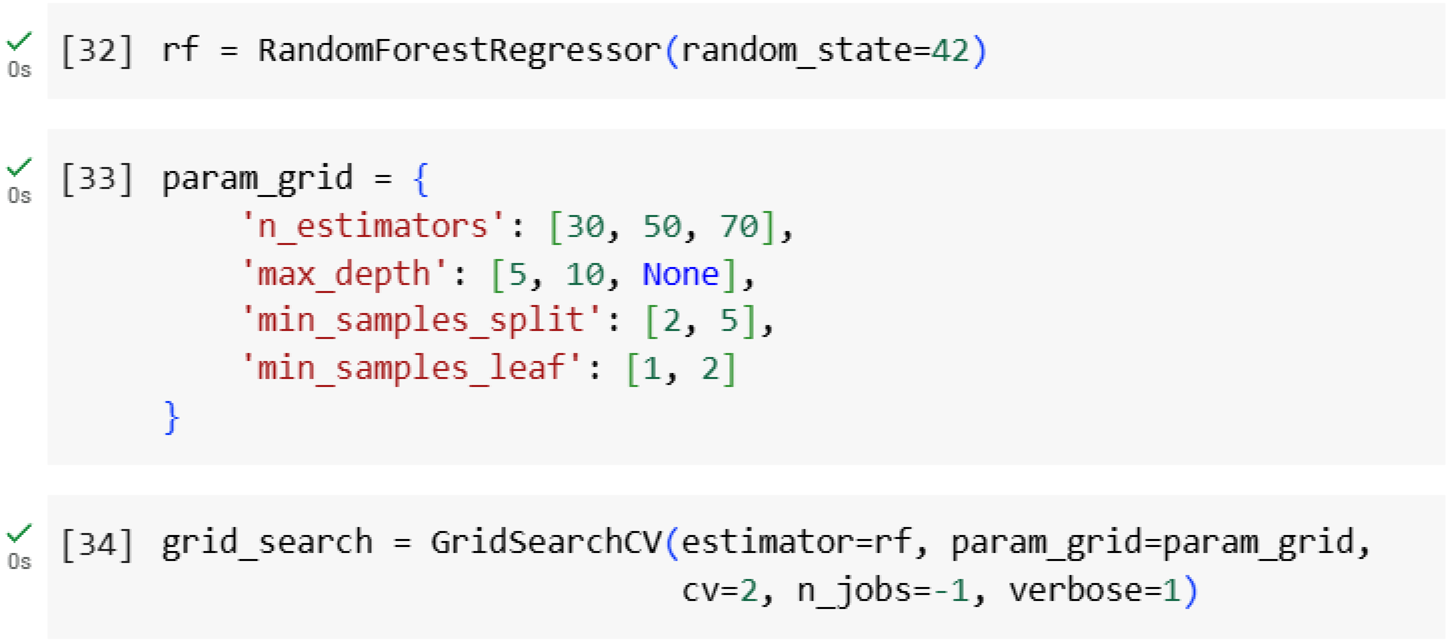
- Used random grid for faster tuning
- After that, adjustments were made manually to ensure maximum efficiency.



```
from sklearn.model_selection import RandomizedSearchCV
xgb_model = XGBRegressor(n_estimators=300)

param_grid = {
    'learning_rate' : [0.01,.03,0.1],
    'max_depth': [6, 10, 15],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0],
    'gamma': [0.1, 0.3, 1],
    'reg_alpha': [0, 0.1, 1.0],
    'reg_lambda': [1.0, 2.5, 5.0],
}
random_search = RandomizedSearchCV(
    estimator=xgb_model,
    param_distributions=param_grid,
    n_iter=100,
    scoring='neg_root_mean_squared_error',
    cv=3,
    verbose=3,
    n_jobs=-1
)
random_search.fit(x_train_scaled, y_train)
```

Hyperparameter Tuning

- Grid Search



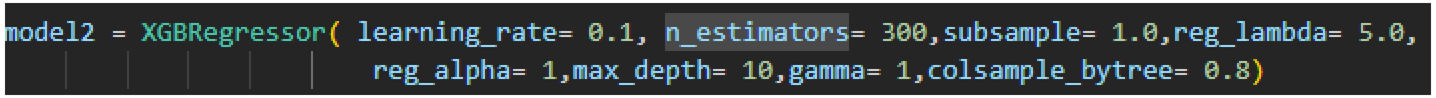
```
✓  [32] rf = RandomForestRegressor(random_state=42)
0s
```

```
✓  [33] param_grid = {
0s              'n_estimators': [30, 50, 70],
                'max_depth': [5, 10, None],
                'min_samples_split': [2, 5],
                'min_samples_leaf': [1, 2]
            }
```

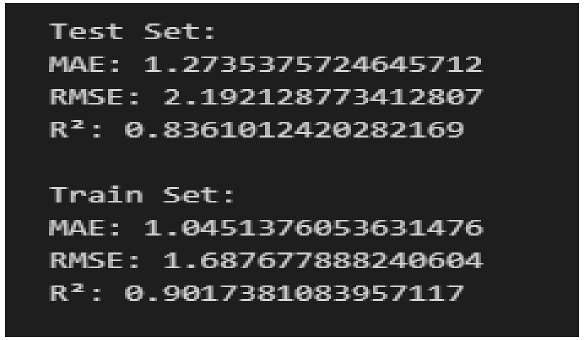
```
✓  [34] grid_search = GridSearchCV(estimator=rf, param_grid=param_grid,
0s                                 cv=2, n_jobs=-1, verbose=1)
```

Best & Evaluation

XGBOOST

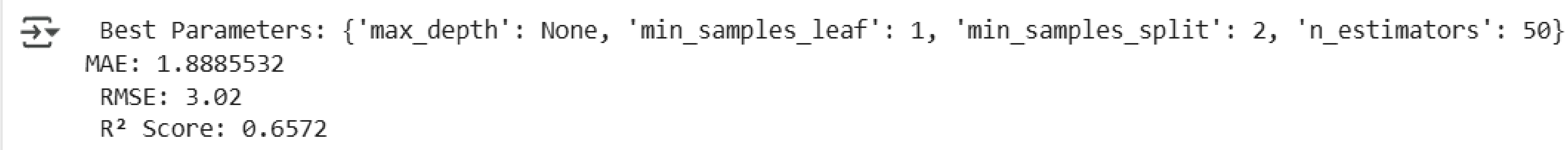


```
model2 = XGBRegressor( learning_rate= 0.1, n_estimators= 300,subsample= 1.0,reg_lambda= 5.0,
                       reg_alpha= 1,max_depth= 10,gamma= 1,colsample_bytree= 0.8)
```



```
Test Set:
MAE: 1.2735375724645712
RMSE: 2.192128773412807
R²: 0.8361012420282169

Train Set:
MAE: 1.0451376053631476
RMSE: 1.687677888240604
R²: 0.9017381083957117
```

Random Forest Grid Search cv2



```
 Best Parameters: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}
MAE: 1.8885532
 RMSE: 3.02
 R² Score: 0.6572
```