Flight Price Prediction

Introduction :

The flight ticket buying system is to purchase a ticket many days prior to flight takeoff so as to stay away from the effect of the most extreme charge. Mostly, aviation routes don't agree this procedure. Plane organizations may diminish the cost at the time, they need to build the market and at the time when the tickets are less accessible. They may maximize the costs. So the cost may rely upon different factors. To foresee the costs this venture uses AI to exhibit the ways of flight tickets after some time. All organizations have the privilege and opportunity to change it's ticket costs at anytime. Explorer can set aside cash by booking a ticket at the least costs. People who had travelled by flight frequently are aware of price fluctuations. The airlines use complex policies of Revenue Management for execution of distinctive evaluating systems.

Definition of the Problem:

It is hard for the client to buy an air ticket at the most reduced cost. For this few procedures are explored to determine time and date to grab air tickets with minimum fare rate. The majority of these systems are utilizing the modern computerized system known as Machine Learning. To determine ideal purchase time for flight ticket Gini for building up a model.

Data Pre-processing Done:

The purpose of preprocessing is to convert raw data into a form that fits machine learning. Structured and clean data allows a data scientist to get

more precise results from an applied machine learning model. The technique includes data formatting, cleaning, and sampling.

- Hardware and Software Requirements and Tools Used

- **Tools: Python 3.8.5, Jupyter Notebook, Numpy, Pandas, Matplotlib, Seaborn, Scikit-learn, Scipy**
- **Techniques: regression, KNeighborsRegressor,Decision Tree Regressor, Lasso, Ridge, RandomForestRegressor**
- **Hardware: I3 processor, 4GB RAM**

- Model/s Development and Evaluation

I have used the below models for regression:

1. Linear Regression
2. Decision Tree Regressor
3. Lasso
4. Ridge
5. RandomForestRegressor

- Identification of possible problem-solving approaches (methods)

- Read the data (from csv)
- Identify the dependent and independent variables.
- Check if the data has missing values or the data is categorical or not.
- If yes, apply basic data preprocessing operations to bring the data in a go to go format.
- Now split the data into the groups of training and testing for the respective purpose.
- After splitting data, fit it to a most suitable model. (How to find a suitable model is answered below)
- Validate the model. If satisfactory, then go with it, else tune the parameters and keep testing. In a few cases, you can also try different algorithms for the same problem to understand the difference between the accuracies.
- From step 7 one can also learn about accuracy paradox.
- Visualize the data.

- Testing of Identified Approaches (Algorithms)

1. Cross validation
2. Linear Regression
3. Decision Tree Regression
4. r2_score
5. mean_squared_error, mean_absolute_error

- Run and Evaluate selected models

I have used the below models for regression:

RandomForestRegressor

```
rdr = RandomForestRegressor()
rdr.fit(x_train,y_train)
y_train_pred = rdr.predict(x_train)
y_test_pred = rdr.predict(x_test)
```

```
print("Train Results for Random Forest Regressor Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_train.values, y_train_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_train.values, y_train_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_train.values, y_train_pred)))
```

```
Train Results for Random Forest Regressor Model:
--------------------------------------------------
Mean_absolute_error 1559.2774729292466
Mean_squared_error 4008082.835792465
R-squared: 2002.0196891620385
```

Ridge

```
rd = Ridge()
rd.fit(x_train,y_train)
y_train_pred = rd.predict(x_train)
y_test_pred = rd.predict(x_test)
```

```
print("Train Results for Ridge Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_train.values, y_train_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_train.values, y_train_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_train.values, y_train_pred)))
```

```
Train Results for Ridge Model:
--------------------------------------------------
Mean_absolute_error 3882.094916325874
Mean_squared_error 23919914.39439255
R-squared: 4890.798952563124
```

Lasso

```
lss = Lasso()
lss.fit(x_train,y_train)
y_train_pred = lss.predict(x_train)
y_test_pred = lss.predict(x_test)
```

```
print("Train Results for Lasso Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_train.values, y_train_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_train.values, y_train_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_train.values, y_train_pred)))
```

```
Train Results for Lasso Model:
--------------------------------------------------
Mean_absolute_error 3882.1335606722987
Mean_squared_error 23919920.761695053
R-squared: 4890.799603510151
```

```
print("Test Results for Lasso Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_test, y_test_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_test, y_test_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_test, y_test_pred)))
```

```
Test Results for Lasso Model:
--------------------------------------------------
Mean_absolute_error 4337.779800498753
Mean_squared_error 31933353.20932668
```

Decision Tree Regressor

```
dtc = DecisionTreeRegressor()
dtc.fit(x_train,y_train)
y_train_pred = dtc.predict(x_train)
y_test_pred = dtc.predict(x_test)
```

```
print("Train Results for Decision Tree Regressor Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_train.values, y_train_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_train.values, y_train_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_train.values, y_train_pred)))
```

```
Train Results for Decision Tree Regressor Model:
--------------------------------------------------
Mean_absolute_error 36.678972712680576
Mean_squared_error 146176.26993044408
R-squared: 382.3300536584118
```

```
print("Test Results for Decision Tree Regressor Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_test, y_test_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_test, y_test_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_test, y_test_pred)))
```

```
Test Results for Decision Tree Regressor Model:
--------------------------------------------------
Mean_absolute_error 4337.779800498753
```

K Neighbors Regressor

```
knn = KNeighborsRegressor()
knn.fit(x_train,y_train)
y_train_pred = knn.predict(x_train)
y_test_pred = knn.predict(x_test)
```

```
print("Train Results for K Neighbors Regressor Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_train.values, y_train_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_train.values, y_train_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_train.values, y_train_pred)))
```

```
Train Results for K Neighbors Regressor Model:
--------------------------------------------------
Mean_absolute_error 3458.1523809523815
Mean_squared_error 19181875.9035206
R-squared: 4379.711851654239
```

```
print("Test Results for K Neighbors Regressor Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_test, y_test_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_test, y_test_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_test, y_test_pred)))
```

```
Test Results for K Neighbors Regressor Model:
--------------------------------------------------
Mean absolute error 4337.779890408753
```

- Conclusion:

The five models produced adequate results on their own, averaging a relatively low RMSE for each dataset. Through this advance algorithm and including pruning and other measures for over fitting, we were able to as a Random Forest Regressor that performed pretty well on their own. These shows that these machine learning techniques can be applied well to predicting sports data. By assembling multiple methods, we where able to create an even more accurate model. Random Forest Regressor is the best model of this dataset, because

Random Forest Regressor has better r2_score than linear regression model , lasso, ridge, Elastic Net for this dataset, r2_score is 82% it may also denote it is over fitting as it even classifies the outliers perfectly.

```
conclusion
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	
predicted	20589.42	20297.39	21496.335	21784.68	19665.615	23147.08	16421.87	23196.07	19715.17	20549.97	18965.23	18055.31	23079.63	20278.43	19846
original	17981.60	17595.40	19898.000	22963.00	19898.000	23402.20	17813.20	22291.00	20084.40	24375.40	21316.00	19586.40	22523.80	18879.00	22362