Doctor's Consultation Fees Prediction

Introduction :

We have all been in a situation where we go to a doctor in an emergency and find that the consultation fees are too high. As a data scientist, we all should do better. In order to achieve our goal, we will use a decision tree. A Random Forest Regressor is well suited to this problem because it can determine and apply rules in order of importance, and is extremely accurate and adaptable. Random Forest Regressor can easily grow with data and can also be easily combined with other techniques for even further accuracy.

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 , Linear Regression , Decision Tree Regressor , KNeighbors Regressor , Random Forest Regressor , SVR.
- 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. Decision Tree Regressor
  4. KNeighbors Regressor
  5. Random Forest Regressor
  6. SVR

- ## Identification of possible problem-solving approaches (methods)

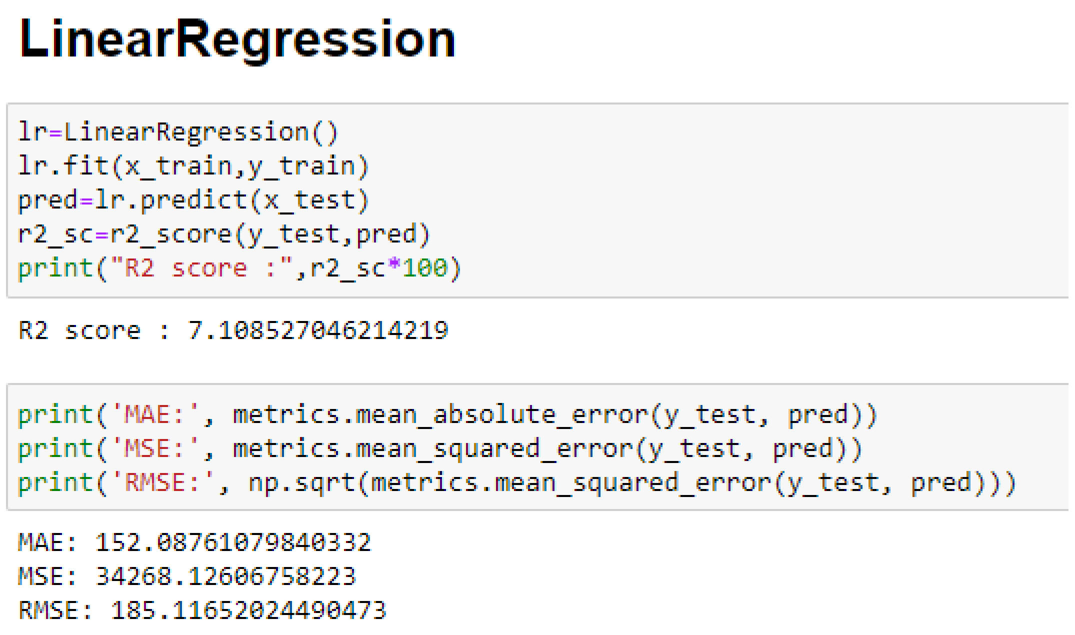
- Read the data (from csv)
- Identify the dependent and independent variables.
- Check if the data has missing values or the data is categorical or not.
- If yes, apply basic data preprocessing operations to bring the data in a go to go format.
- Now split the data into the groups of training and testing 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)

- Cross validation
- Linear Regression
- Decision Tree Regression
- r2_score
- mean_squared_error, mean_absolute_error

- Run and Evaluate selected models:

I have used the below models for regression:



LinearRegression

```
lr=LinearRegression()
lr.fit(x_train,y_train)
pred=lr.predict(x_test)
r2_sc=r2_score(y_test,pred)
print("R2 score :",r2_sc*100)
```

```
R2 score : 7.108527046214219
```

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

```
MAE: 152.08761079840332
MSE: 34268.12606758223
RMSE: 185.11652024490473
```

DecisionTreeRegressor

```
from sklearn.tree import DecisionTreeRegressor
dtr=DecisionTreeRegressor()
dtr.fit(x_train,y_train)
pred=dtr.predict(x_test)
r2_sc=r2_score(y_test,pred)
print("R2 score :",r2_sc*100)
```

```
R2 score : -53.42108419624401
```

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

```
MAE: 172.39396479463537
MSE: 56597.80050293378
RMSE: 237.902922434622
```

KNeighborsRegressor

```
knn=KNeighborsRegressor()
knn.fit(x_train,y_train)
pred=knn.predict(x_test)
r2_sc=r2_score(y_test,pred)
print("R2 score :",r2_sc*100)
```

```
R2 score : 0.34435733326221296
```

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

```
MAE: 150.8601844090528
MSE: 36763.46189438391
RMSE: 191.7380032606575
```

RandomForestRegressor

```
rdr = RandomForestRegressor()
rdr.fit(x_train,y_train)
pred=rdr.predict(x_test)
r2_sc=r2_score(y_test,pred)
print("R2 score :",r2_sc*100)
```

```
R2 score : 12.592687793563295
```

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

```
MAE: 140.2633952021714
MSE: 32244.99191017101
RMSE: 179.56890574420453
```

SVR

```
svr = SVR()
svr.fit(x_train,y_train)
pred=svr.predict(x_test)
r2_sc=r2_score(y_test,pred)
print("R2 score :",r2_sc*100)
```

```
R2 score : 4.0469878832906865
```

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

```
MAE: 148.2292288018253
MSE: 35397.54308143557
RMSE: 188.1423479215553
```

- Conclusion:

The five models produced adequate results on their own, averaging a relatively low RMSE for each data set. Through these advanced algorithms 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. This shows that these machine learning techniques can be applied well to predicting Doctors fees consultation data. By assembling multiple methods, we were able to create an even more accurate model. Random Forest Regressor is the best model of this dataset, because Random Forest Regressor is accuracy score and cross val score is good than other models.