Overview

The project “Doctor’s Fee Prediction” has been undertaken based on an online live hackathon of machinehack.com. The website aims to generate efficient mechanisms for handling, management and analysis of real time data. The dataset of doctor’s features was provided by the hackathon organizers and approximately 30 days were given in order to achieve the goal.

We have all been in situation where we go to a doctor in emergency and find that the consultation fees are too high. As a data scientist we all should do better. What if you have data that records important details about a doctor and you get to build a model to predict the doctor’s consulting fee.? This is the hackathon that lets you do that.

Thus, the project basically aims to predict the consultation fee of a doctor, given his/her features. The provided dataset was subjected to internal modifications like normalization, standardization, etc. and various machine learning algorithms have been used to test the prediction capacity as well as overall accuracy.

The entire project has been implemented in Python 2.7 programming language due to set of its powerful machine learning libraries and significant characteristics offered by it and none of the paid libraries or ready-made materials were used for implementation of this project.

At the end of the project, the obtained accuracies of different prediction algorithms have been compared.

List of undertaken steps

It was decided to divide the project into parts due to complexity of given dataset and comparative analysis of various algorithms. Thus, the overall workflow of this project has taken a pipeline shape, where each step takes input as the output of the previous step and its own output is used as an input to the next step. Following steps were undertaken and finally merged to complete this project:

1] Cleaning the given dataset, and converting the full sentences into a form which can be inputted to any prediction algorithm.

2] Inputting the missing or unknown values by “Smoothing by means” technique.

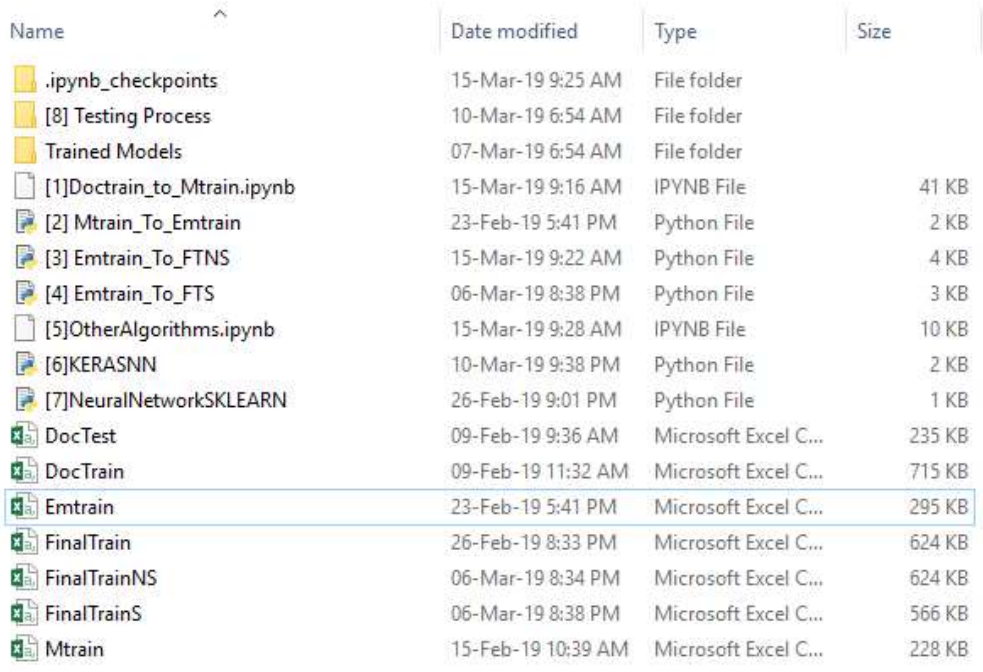
3] Encoding the dataset in order to convert every character value into a numerical value which is to be further inputted into prediction algorithms.

4] Training various prediction algorithms like Decision Tree Regressor and Polynomial Regression.

5] Training a multilayer perceptron and ANN for more accurate predictions.

6] Converting the Test dataset as per steps 1,2 and 3.

7] Testing the predictions as per all of the trained models.



Name	Date modified	Type	Size
.ipynb_checkpoints	15-Mar-19 9:25 AM	File folder	
[8] Testing Process	10-Mar-19 6:54 AM	File folder	
Trained Models	07-Mar-19 6:54 AM	File folder	
[1]Doctrain_to_Mtrain.ipynb	15-Mar-19 9:16 AM	IPYNB File	41 KB
[2] Mtrain_To_Emtrain	23-Feb-19 5:41 PM	Python File	2 KB
[3] Emtrain_To_FTNS	15-Mar-19 9:22 AM	Python File	4 KB
[4] Emtrain_To_FTS	06-Mar-19 8:38 PM	Python File	3 KB
[5]OtherAlgorithms.ipynb	15-Mar-19 9:28 AM	IPYNB File	10 KB
[6]KERASNN	10-Mar-19 9:38 PM	Python File	2 KB
[7]NeuralNetworkSKLEARN	26-Feb-19 9:01 PM	Python File	1 KB
DocTest	09-Feb-19 9:36 AM	Microsoft Excel C...	235 KB
DocTrain	09-Feb-19 11:32 AM	Microsoft Excel C...	715 KB
Emtrain	23-Feb-19 5:41 PM	Microsoft Excel C...	295 KB
FinalTrain	26-Feb-19 8:33 PM	Microsoft Excel C...	624 KB
FinalTrainNS	06-Mar-19 8:34 PM	Microsoft Excel C...	624 KB
FinalTrainS	06-Mar-19 8:38 PM	Microsoft Excel C...	566 KB
Mtrain	15-Feb-19 10:39 AM	Microsoft Excel C...	228 KB

Figure 1 All of the project Files

About the given dataset

Size of training set: 5961 records

Size of test set: 1987 records

FEATURES:

Qualification: Qualification and degrees held by the doctor

Experience: Experience of the doctor in number of years

Rating: Rating given by patients

Profile: Type of the doctor

Miscellaeous_Info: Extra information about the doctor

Fees: Fees charged by the doctor

Place: Area and the city where the doctor is located.

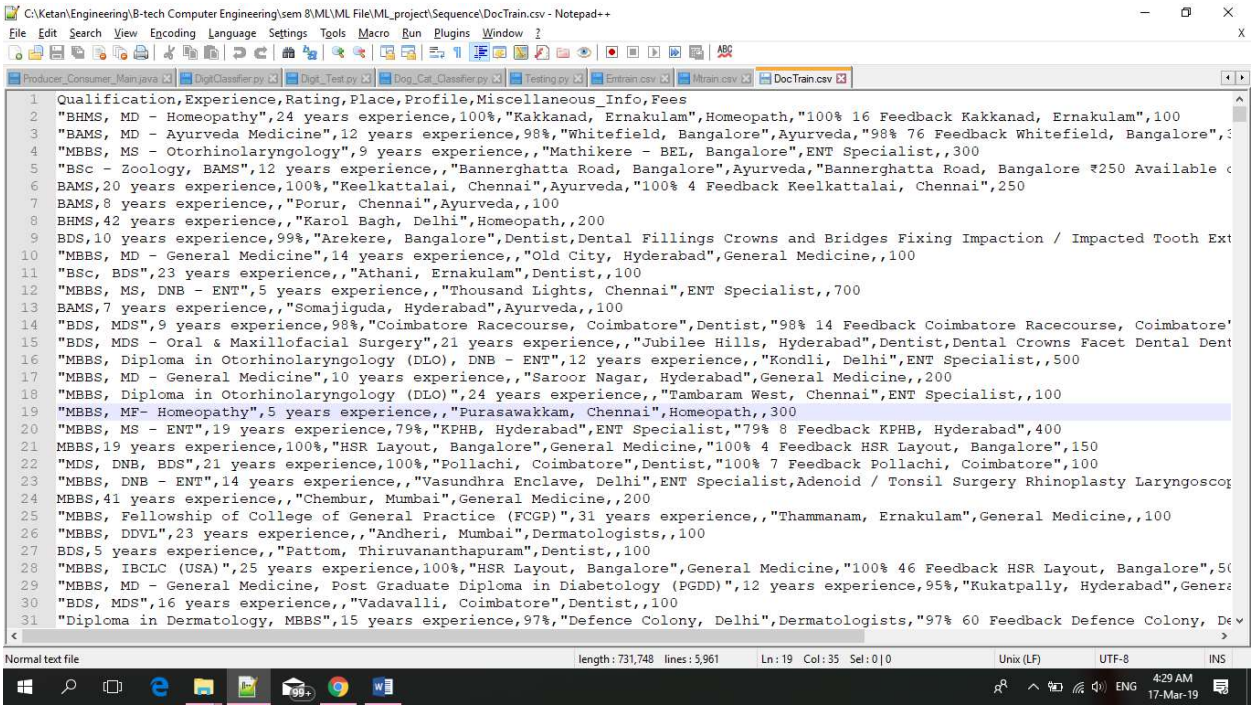
Submissions are evaluated on Root-Mean-Squared-Error (RMSE) between the predicted value and observed score values. The final score calculation is done in the following way:

Submissions are evaluated on Root-Mean-Squared-Log-Error (RMSLE) error = RMSLE (error)

Score = 1 – error

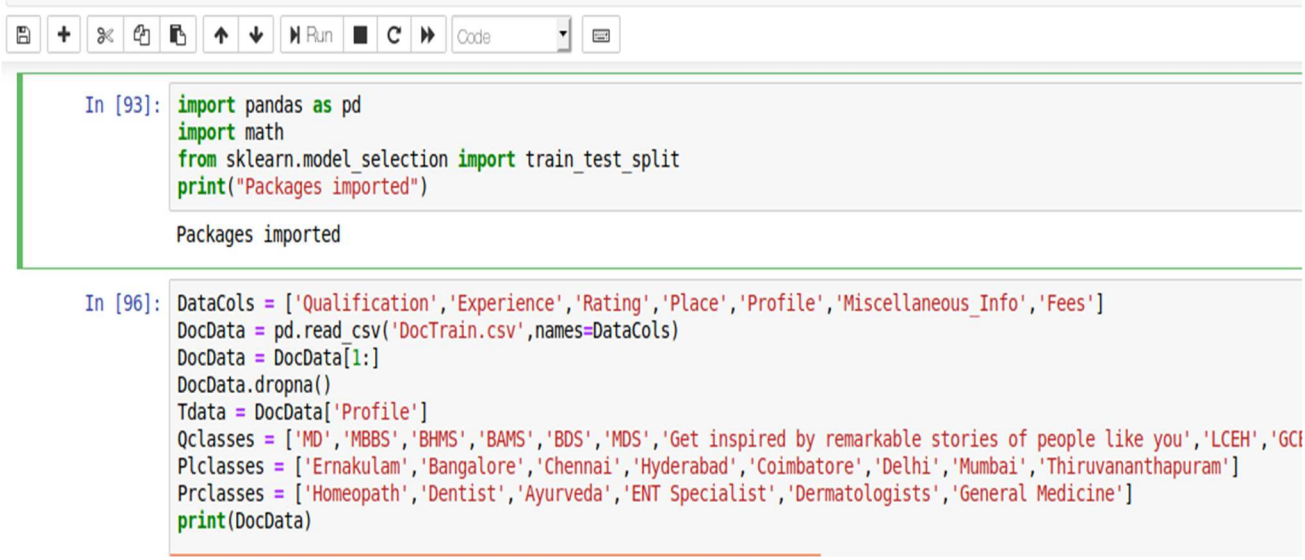
[1] Cleaning the given dataset (1st clean)

Given dataset's snapshot:



```
Qualification,Experience,Rating,Place,Profile,Miscellaneous_Info,Fees
"BHMS, MD - Homeopathy",24 years experience,100%,"Kakkanad, Ernakulam",Homeopath,"100% 16 Feedback Kakkanad, Ernakulam",100
"BAMS, MD - Ayurveda Medicine",12 years experience,98%,"Whitefield, Bangalore",Ayurveda,"98% 76 Feedback Whitefield, Bangalore",
"MBBS, MS - Otorhinolaryngology",9 years experience,,"Mathikere - BEL, Bangalore",ENT Specialist,,300
"BSc - Zoology, BAMS",12 years experience,,"Bannerghatta Road, Bangalore",Ayurveda,"Bannerghatta Road, Bangalore ₹250 Available
BAMS,20 years experience,100%,"Keelkattalai, Chennai",Ayurveda,"100% 4 Feedback Keelkattalai, Chennai",250
BAMS,8 years experience,,"Porur, Chennai",Ayurveda,,100
BHMS,42 years experience,,"Karol Bagh, Delhi",Homeopath,,200
BDS,10 years experience,99%,"Arekere, Bangalore",Dentist,Dental Fillings Crowns and Bridges Fixing Impaction / Impacted Tooth Ext
"MBBS, MD - General Medicine",14 years experience,,"Old City, Hyderabad",General Medicine,,100
"BSc, BDS",23 years experience,,"Athani, Ernakulam",Dentist,,100
"MBBS, MS, DNB - ENT",5 years experience,,"Thousand Lights, Chennai",ENT Specialist,,700
BAMS,7 years experience,,"Somajiguda, Hyderabad",Ayurveda,,100
"BDS, MDS",9 years experience,98%,"Coimbatore Racecourse, Coimbatore",Dentist,"98% 14 Feedback Coimbatore Racecourse, Coimbatore'
"BDS, MDS - Oral & Maxillofacial Surgery",21 years experience,,"Jubilee Hills, Hyderabad",Dentist,Dental Crowns Facet Dental Dent
"MBBS, Diploma in Otorhinolaryngology (DLO), DNB - ENT",12 years experience,,"Kondli, Delhi",ENT Specialist,,500
"MBBS, MD - General Medicine",10 years experience,,"Saroor Nagar, Hyderabad",General Medicine,,200
"MBBS, Diploma in Otorhinolaryngology (DLO)",24 years experience,,"Tambaram West, Chennai",ENT Specialist,,100
"MBBS, MF- Homeopathy",5 years experience,,"Purasawakkam, Chennai",Homeopath,,300
"MBBS, MS - ENT",19 years experience,79%,"KPHB, Hyderabad",ENT Specialist,"79% 8 Feedback KPHB, Hyderabad",400
MBBS,19 years experience,100%,"HSR Layout, Bangalore",General Medicine,"100% 4 Feedback HSR Layout, Bangalore",150
"MDS, DNB, BDS",21 years experience,100%,"Pollachi, Coimbatore",Dentist,"100% 7 Feedback Pollachi, Coimbatore",100
"MBBS, DNB - ENT",14 years experience,,"Vasundhra Enclave, Delhi",ENT Specialist,Adenoid / Tonsil Surgery Rhinoplasty Laryngosco
MBBS,41 years experience,,"Chembur, Mumbai",General Medicine,,200
"MBBS, Fellowship of College of General Practice (FCGP)",31 years experience,,"Thammanam, Ernakulam",General Medicine,,100
"MBBS, DDVL",23 years experience,,"Andheri, Mumbai",Dermatologists,,100
BDS,5 years experience,,"Pattom, Thiruvananthapuram",Dentist,,100
"MBBS, IBCLC (USA)",25 years experience,100%,"HSR Layout, Bangalore",General Medicine,"100% 46 Feedback HSR Layout, Bangalore",5
"MBBS, MD - General Medicine, Post Graduate Diploma in Diabetology (PGDD)",12 years experience,95%,"Kukatpally, Hyderabad",Genera
"BDS, MDS",16 years experience,,"Vadavalli, Coimbatore",Dentist,,100
"Diploma in Dermatology, MBBS",15 years experience,97%,"Defence Colony, Delhi",Dermatologists,"97% 60 Feedback Defence Colony, De
```

It can be clearly seen that this complex format of given dataset cannot be inputted to any prediction algorithms. Thus, cleaning process was preformed as follows in order to mold this dataset as per requirement:



```
In [93]: import pandas as pd
         import math
         from sklearn.model_selection import train_test_split
         print("Packages imported")

         Packages imported

In [96]: DataCols = ['Qualification','Experience','Rating','Place','Profile','Miscellaneous_Info','Fees']
         DocData = pd.read_csv('DocTrain.csv',names=DataCols)
         DocData = DocData[1:]
         DocData.dropna()
         Tdata = DocData['Profile']
         Qclasses = ['MD','MBBS','BHMS','BAMS','BDS','MDS','Get inspired by remarkable stories of people like you','LCEH','GCI
         Plclasses = ['Ernakulam','Bangalore','Chennai','Hyderabad','Coimbatore','Delhi','Mumbai','Thiruvananthapuram']
         Prclasses = ['Homeopath','Dentist','Ayurveda','ENT Specialist','Dermatologists','General Medicine']
         print(DocData)
```

	Qualification	Experience	\
1	BHMS, MD - Homeopathy	24 years experience	
2	BAMS, MD - Ayurveda Medicine	12 years experience	
3	MBBS, MS - Otorhinolaryngology	9 years experience	
4	BSc - Zoology, BAMS	12 years experience	
5	BAMS	20 years experience	
6	BAMS	8 years experience	
7	BHMS	42 years experience	
8	BDS	10 years experience	
9	MBBS, MD - General Medicine	14 years experience	
10	BSc, BDS	23 years experience	
11	MBBS, MS, DNB - ENT	5 years experience	
12	BAMS	7 years experience	
13	BDS, MDS	9 years experience	
14	BDS, MDS - Oral & Maxillofacial Surgery	21 years experience	
15	MBBS, Diploma in Otorhinolaryngology (DLO), DN...	12 years experience	
16	MBBS, MD - General Medicine	10 years experience	
17	MBBS, Diploma in Otorhinolaryngology (DLO)	24 years experience	
18	MBBS, MF- Homeopathy	5 years experience	

In [97]:
```
# Molding the dataset
QUALS = []
Tdata = DocData['Qualification'].values
Tdata = list(Tdata)

for sample in Tdata:
    Include = []
    for qual in Qclasses:
        if qual in sample:
            Include.append(qual)

    NewText = ""
    for I in Include:
        NewText = NewText + I + ","

    NewText = NewText[:-1]
    QUALS.append(NewText)

QUALS_F = []
for row in QUALS:
    if row=='':
        QUALS_F.append('MS')
    else:
        QUALS_F.append(row)

# QUALS_F contains the FInal Qualifications to be appended in new dataframe
```

```
DocPlaces = []
Tdata = DocData['Place'].values
Tdata = list(Tdata)

for sample in Tdata:
    if sample!=sample:
        DocPlaces.append('Unknown')
    else:
        f=0
        for place in Plclasses:
            if place in sample:
                DocPlaces.append(place)
                f=1
                break

        if f==0:
            DocPlaces.append('Unknown')


#DocPlaces contains Places of Doctors to be appended
```

In [15]:
```
DocProfiles = []
Tdata = DocData['Profile'].values
Tdata = list(Tdata)

DocProfiles = Tdata
#Doc Profiles are finalized Profiles
```

In [ ]:
```

```

In [ ]:
```

```

In [99]:
```
Ttdata=DocData['Fees'].values
print(Ttdata)
Fdata = []
for i in Ttdata:
    i = int(i)
    Fdata.append(i)
#Now, we have integers in Fdata list as our Fees
```

```
['100' '350' '300' ... '600' '100' '200']
```

In [101]:
```
Tdata = DocData['Experience'].values
Tdata = list(Tdata)
ExpData = []

for sample in Tdata:
    sample = sample[:-17]
    temp = int(sample)
    ExpData.append(temp)

    # ExpData stores Exps to be appended
```

In [103]:
```
#Now, Creating a new DataFrame
NewDocData = []
DataCols = ['Qualification','Experience','Rating','Place','Profile','Fees']
for i in range(len(Tdata)):
    temp = []
    temp.append(QUALS_F[i])
    temp.append(ExpData[i])
    temp.append(RatData[i])
    temp.append(DocPlaces[i])
    temp.append(DocProfiles[i])
    temp.append(Fdata[i])
    NewDocData.append(temp)


myFrame = pd.DataFrame(NewDocData,columns=DataCols)
myFrame.to_csv('Mtrain.csv')
print(myFrame)
```

Obtained CSV file:-

```
,Qualification,Experience,Rating,Place,Profile,Fees
0,"MD,BHMS",24,100,Ernakulam,Unknown,100
1,"MD,BAMS",12,98,Bangalore,Unknown,350
2,MBBS,9,0,Bangalore,Unknown,300
3,BAMS,12,0,Bangalore,Unknown,250
4,BAMS,20,100,Chennai,Unknown,250
5,BAMS,8,0,Chennai,Unknown,100
6,BHMS,42,0,Delhi,Unknown,200
7,BDS,10,99,Bangalore,Unknown,200
8,"MD,MBBS",14,0,Hyderabad,Unknown,100
9,BDS,23,0,Ernakulam,Unknown,100
10,"MBBS,DNB",5,0,Chennai,Unknown,700
11,BAMS,7,0,Hyderabad,Unknown,100
12,"MD,BDS,MDS",9,98,Coimbatore,Unknown,200
13,"MD,BDS,MDS",21,0,Hyderabad,Unknown,350
14,"MBBS,DNB,DLO",12,0,Delhi,Unknown,500
15,"MD,MBBS",10,0,Hyderabad,Unknown,200
16,"MBBS,DLO",24,0,Chennai,Unknown,100
17,MBBS,5,0,Chennai,Unknown,300
18,MBBS,19,79,Hyderabad,Unknown,400
19,MBBS,19,100,Bangalore,Unknown,150
20,"MD,BDS,MDS,DNB",21,100,Coimbatore,Unknown,100
```

[2] Filling up the missing and unknown cells

The obtained dataset from step 1 contained many unknown as well as missing values. Thus, it was subjected to smoothing process as follows for imputation of missing values:

```
import pandas as pd
from sklearn.preprocessing import Imputer
import numpy as np




DATA1 = pd.read_csv('Mtrain.csv')
DATA2 = pd.read_csv('DocTrain.csv')

Quals = DATA1.iloc[:,1].values

for v in range(len(Quals)):
    if "MD" in Quals[v] or "M.D." in Quals[v]:
        Quals[v]="MD"


for v in range(len(Quals)):
    if "MBBS" in Quals[v]:
        Quals[v]="MBBS"


for v in range(len(Quals)):
    if "MDS" in Quals[v]:
        Quals[v]="MDS"

for v in range(len(Quals)):
    if "DNB" in Quals[v]:
        Quals[v]="DNB"
```

```
for v in range(len(Quals)):
    if "DNB" in Quals[v]:
        Quals[v]="DNB"

for v in range(len(Quals)):
    if "BAMS" in Quals[v]:
        Quals[v]="BAMS"

for v in range(len(Quals)):
    if "BDS" in Quals[v]:
        Quals[v]="BDS"

for v in range(len(Quals)):
    if "BHMS" in Quals[v]:
        Quals[v]="BHMS"

for v in range(len(Quals)):
    if "DDVL" in Quals[v]:
        Quals[v]="DDVL"

DATA1.iloc[:,1] = Quals
#Done with the Qualifications

ExpData = DATA1.iloc[:,2].values
imputer = Imputer(missing_values = 0, strategy = "mean", axis = 0)
ExpData = np.array(ExpData).reshape(-1,1)
imputer = imputer.fit(ExpData)
ExpData= imputer.transform(ExpData)
DATA1.iloc[:,2] = ExpData
#Done with the Experience
```

```
RatData = DATA1.iloc[:,3].values
imputer = Imputer(missing_values = 0, strategy = "mean", axis = 0)
RatData = np.array(RatData).reshape(-1,1)
imputer = imputer.fit(RatData)
RatData= imputer.transform(RatData)
DATA1.iloc[:,3] = RatData
#Done with the Ratings

PlData = DATA1.iloc[:,4].values
# Done with the Places

ProfData = DATA2.iloc[:,4].values
#Done with the Profiles

Fees = DATA1.iloc[:,-1].values
#Done with the Fees


Flist = []

for index in range(len(Quals)):
    temp = []
    temp.append(Quals[index])
    temp.append(ExpData[index][0])
    temp.append(RatData[index][0])
    temp.append(PlData[index])
    temp.append(ProfData[index])
    temp.append(Fees[index])

    Flist.append(temp)
```

```
for index in range(len(Quals)):
    temp = []
    temp.append(Quals[index])
    temp.append(ExpData[index][0])
    temp.append(RatData[index][0])
    temp.append(PlData[index])
    temp.append(ProfData[index])
    temp.append(Fees[index])

    Flist.append(temp)


Cols = ['Qualification','Experiece','Ratings','Place','Profile','Fees']
df = pd.DataFrame(Flist, columns = Cols)

df.to_csv('Emtrain.csv')
```

Obtained CSV file:

```
,Qualification,Experiece,Ratings,Place,Profile,Fees
0,MD,24.0,100.0,Ernakulam,Homeopath,100
1,MD,12.0,98.0,Bangalore,Ayurveda,350
2,MBBS,9.0,94.6425884123401,Bangalore,ENT Specialist,300
3,BAMS,12.0,94.6425884123401,Bangalore,Ayurveda,250
4,BAMS,20.0,100.0,Chennai,Ayurveda,250
5,BAMS,8.0,94.6425884123401,Chennai,Ayurveda,100
6,BHMS,42.0,94.6425884123401,Delhi,Homeopath,200
7,BDS,10.0,99.0,Bangalore,Dentist,200
8,MD,14.0,94.6425884123401,Hyderabad,General Medicine,100
9,BDS,23.0,94.6425884123401,Ernakulam,Dentist,100
10,MBBS,5.0,94.6425884123401,Chennai,ENT Specialist,700
11,BAMS,7.0,94.6425884123401,Hyderabad,Ayurveda,100
12,MD,9.0,98.0,Coimbatore,Dentist,200
13,MD,21.0,94.6425884123401,Hyderabad,Dentist,350
14,MBBS,12.0,94.6425884123401,Delhi,ENT Specialist,500
15,MD,10.0,94.6425884123401,Hyderabad,General Medicine,200
16,MBBS,24.0,94.6425884123401,Chennai,ENT Specialist,100
17,MBBS,5.0,94.6425884123401,Chennai,Homeopath,300
18,MBBS,19.0,79.0,Hyderabad,ENT Specialist,400
19,MBBS,19.0,100.0,Bangalore,General Medicine,150
20,MD,21.0,100.0,Coimbatore,Dentist,100
21,MBBS,14.0,94.6425884123401,Delhi,ENT Specialist,500
22,MBBS,41.0,94.6425884123401,Mumbai,General Medicine,200
23,MBBS,31.0,94.6425884123401,Ernakulam,General Medicine,100
24,MBBS,23.0,94.6425884123401,Mumbai,Dermatologists,100
25,BDS,5.0,94.6425884123401,Thiruvananthapuram,Dentist,100
```

[3] Encoding of dataset

Although the missing values were imputed and dataset was precleaned, the textual sentences were still needed to be converted into numerical values for more accurate predictions. Thus, the obtained csv from step 2 was subjected to following process: -

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
import numpy as np
DATA = pd.read_csv('Emtrain.csv')

Quals = DATA.iloc[:,1].values
Exps= DATA.iloc[:,2].values
Ratings = DATA.iloc[:,3].values
Places = DATA.iloc[:,4].values
Profiles = DATA.iloc[:,5].values
Fees = DATA.iloc[:,-1].values
sc = StandardScaler()
# Encoding Qualifications:
MAP = {}
MAP['MD'] = 180
MAP['MBBS'] = 170
MAP['BDS'] = 70
MAP['G.A.M.S'] = 50
MAP['BHMS'] = 80
MAP['Get inspired by remarkable stories of people like you'] = 20
MAP['DLO'] = 150
MAP['PhD'] = 140
MAP['DDVL'] = 60
MAP['DDV'] = 40
MAP['MS'] = 190
MAP['DNB'] = 110
MAP['Diploma in Dermatology'] = 30
MAP['LCEH'] = 100
MAP['GCEH'] = 90
MAP['BSAM'] = 160
```

```
Qualsn = []

for i in range(len(Quals)):
    temp = Quals[i]
    ans = MAP[temp]
    Qualsn.append(ans)



print(Qualsn[:10])
Qualsns = np.array(Qualsn).reshape(-1,1)
scaler = sc.fit(Qualsns)
QualsS = scaler.fit_transform(Qualsns)
# Encoding of Qualifications Finished!


# Encoding Places
MAP = {}
MAP['Ernakulam'] = 80
MAP['Chennai'] = 60
MAP['Coimbatore'] = 70
MAP['Unknown'] = 30
MAP['Delhi'] = 90
MAP['Thiruvananthapuram'] = 50
MAP['Mumbai'] = 120
MAP['Bangalore'] = 110
MAP['Hyderabad'] = 100

Placesn = []
```

```
Placesn = []

for i in range(len(Places)):
    temp = Places[i]
    ans = MAP[temp]
    Placesn.append(ans)



print(Placesn[:10])
Placesns = np.array(Placesn).reshape(-1,1)
scaler = sc.fit(Placesns)
PlacesS = scaler.fit_transform(Placesns)
# Encoding of Places Done!


#Now, Encoding the Profiles

MAP = {}
MAP['Homeopath'] = 70
MAP['Ayurveda'] = 130
MAP['General Medicine'] = 150
MAP['Dentist'] = 100
MAP['Dermatologists'] = 110
MAP['ENT Specialist'] = 170

Profilesn= []
```

```
Profilesn= []

for i in range(len(Profiles)):
    temp = Profiles[i]
    ans = MAP[temp]
    Profilesn.append(ans)


print(Profilesn[:10])
Profilesns = np.array(Profilesn).reshape(-1,
scaler = sc.fit(Profilesns)
ProfilesS = scaler.fit_transform(Profilesns)
# Profiles encoded Too!


Expsns = np.array(Exps).reshape(-1,1)
scaler = sc.fit(Expsns)
ExpsS = scaler.fit_transform(Expsns)



Ratingsns = np.array(Ratings).reshape(-1,1)
scaler = sc.fit(Ratingsns)
RatingS = scaler.fit_transform(Ratingsns)

Flist = []
```

```
for index in range(len(Quals)):
    temp = []
    temp.append(QualsS[index][0])
    temp.append(ExpsS[index][0])
    temp.append(RatingS[index][0])
    temp.append(PlacesS[index][0])
    temp.append(ProfilesS[index][0])
    temp.append(Fees[index])

    Flist.append(temp)

Cols = ['Qualification','Experiece','Ratings','Place','Profile','Fees']
df = pd.DataFrame(Flist, columns = Cols)

df.to_csv('FinalTrainS.csv')
```

Obtained CSV file:

```
,Qualification,Experiece,Ratings,Place,Profile,Fees
0,0.7379178707284019,0.5890873520558627,1.0215186995749257,0.7379178707284021,0.7379178707284023,100
1,0.7379178707284019,-0.507885459129769,0.6401708479639499,0.7379178707284021,0.7379178707284023,350
2,0.5030846215070107,-0.7821286619261769,-1.968722418411772e-15,0.5030846215070109,0.503084621507011,300
3,-0.6710816245999445,-0.507885459129769,-1.968722418411772e-15,-0.6710816245999442,-0.671081624599944,250
4,-0.6710816245999445,0.22342974832731882,1.0215186995749257,-0.6710816245999442,-0.671081624599944,250
5,-0.6710816245999445,-0.8735430628583128,-1.968722418411772e-15,-0.6710816245999442,-0.671081624599944,100
6,-1.6104146214855088,2.2345465688343102,-1.968722418411772e-15,-1.6104146214855082,-1.6104146214855082,200
7,-1.8452478707069,-0.6907142609940409,0.8308447737694378,-1.8452478707068993,-1.845247870706899,200
8,0.7379178707284019,-0.32505665726549704,-1.968722418411772e-15,0.7379178707284021,0.7379178707284023,100
9,-1.8452478707069,0.4976729511237267,-1.968722418411772e-15,-1.8452478707068993,-1.845247870706899,100
10,0.5030846215070107,-1.1477862656547209,-1.968722418411772e-15,0.5030846215070109,0.503084621507011,700
11,-0.6710816245999445,-0.9649574637904488,-1.968722418411772e-15,-0.6710816245999442,-0.671081624599944,100
12,0.7379178707284019,-0.7821286619261769,0.6401708479639499,0.7379178707284021,0.7379178707284023,200
```

[4] Decision Tree Regressor and Polynomial Regression.

In [1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn import metrics
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import accuracy_score
import pickle
from sklearn.metrics import mean_squared_error

print("All the required packages imported!")
```

```
All the required packages imported!
```

In [3]:
```
# Polynomial Regression
DATA = pd.read_csv('FinalTrain.csv')
X = DATA.iloc[:,1:-1].values
Y = DATA.iloc[:,-1].values
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
poly = PolynomialFeatures(degree=5)
poly_x=poly.fit_transform(X_train)
reg = LinearRegression(normalize=True)
reg.fit(poly_x,Y_train)
Ypred = reg.predict(poly.fit_transform(X_test))
df = pd.DataFrame({'Actual': Y_test, 'Predicted': Ypred})
MSE = mean_squared_error(Y_test,Ypred)
print(df[0:15])
print("The Mean squared error is "+str(MSE))
```

```
    Actual   Predicted
0      150  289.767688
1       50  337.892688
2      400  265.267688
3      250  347.486438
4      300  337.892688
5      200  236.892688
6      200  325.267688
7      300  337.892688
8      200  236.017688
9      700  334.111438
10     300  342.267688
11     500 -239.232312
12     100  308.486438
13     200  224.767688
14     100  375.705188
The Mean squared error is 32938.358165264115
```

In [4]:
```
# Decision Tree
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor()
regressor.fit(X_train, Y_train)
y_pred = regressor.predict(X_test)
df=pd.DataFrame({'Actual':Y_test, 'Predicted':y_pred})
MSE = mean_squared_error(Y_test,Ypred)
print(df[0:15])
print("The Mean squared error is "+str(MSE))
```

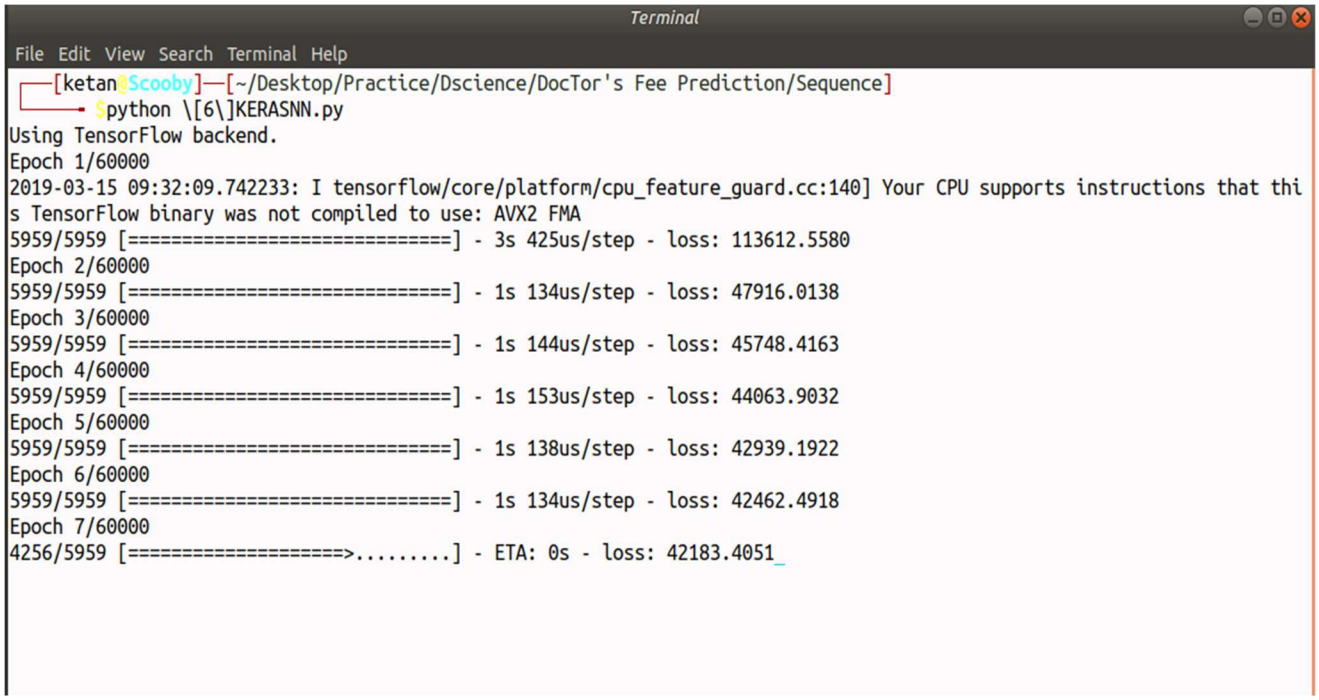
```
    Actual   Predicted
0      150  328.181818
1       50  303.773585
2      400  300.000000
3      250  341.428571
4      300  303.773585
5      200  178.500000
6      200  311.538462
7      300  303.773585
8      200  156.666667
9      700  258.500000
10     300  283.870968
11     500  500.000000
12     100  356.666667
13     200  150.000000
14     100  500.000000
The Mean squared error is 32938.358165264115
```

5] Training a multilayer perceptron and ANN for more accurate predictions.

```
# Multilayer Perceptron
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
mlp = MLPClassifier(hidden_layer_sizes=(32,64,32))
mlp.fit(X_train,Y_train)
Ypred = mlp.predict(X_test)
df=pd.DataFrame({'Actual':Y_test, 'Predicted':Ypred})
print(df[0:15])
MSE = mean_squared_error(Y_test,Ypred)
print("The Mean squared error is "+str(MSE))
print("The r2 value is "+str(r2_score(Y_test,Ypred)))
```

```
    Actual  Predicted
0      150        100
1       50        100
2      400        200
3      250        700
4      300        250
5      200        300
6      200        100
7      300        300
8      200        200
9      700        400
10     300        500
11     500        500
12     100        400
13     200        200
14     100        700
The Mean squared error is 46412.6677852349
The r2 value is -0.3030338761301421
```

```
import numpy as np
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense,Dropout
from keras.layers import LeakyReLU,Dropout
from keras.optimizers import Adam

DATA = pd.read_csv('FinalTrainS.csv')

X = DATA.iloc[:,1:-1].values
Y = DATA.iloc[:,-1].values

model = Sequential()
model.add(Dense(50, input_dim=5, activation='tanh')
model.add(Dense(50))
model.add(LeakyReLU(alpha=0.2))
model.add(Dense(75))
model.add(LeakyReLU(alpha=0.1))
model.add(Dense(75))
model.add(LeakyReLU(alpha=0.2))
model.add(Dropout(0.5))
model.add(Dense(100))
model.add(LeakyReLU(alpha=0.1))
model.add(Dense(100))
model.add(LeakyReLU(alpha=0.1))
model.add(Dropout(0.5))
model.add(Dense(75))
model.add(LeakyReLU(alpha=0.1))
model.add(Dense(75))
model.add(LeakyReLU(alpha=0.1))
model.add(Dropout(0.5))
model.add(Dense(50))
model.add(LeakyReLU(alpha=0.1))
model.add(Dense(50))
model.add(LeakyReLU(alpha=0.2))
model.add(Dense(50))
model.add(LeakyReLU(alpha=0.1))
model.add(Dropout(0.5))
model.add(Dense(25))
model.add(LeakyReLU(alpha=0.1))
model.add(Dense(25))
model.add(LeakyReLU(alpha=0.1))
model.add(Dense(1,kernel_initializer='normal'))
model.compile(loss='mean_squared_error', optimizer=Adam(lr=0.0001))
model.fit(X,Y, epochs=60000)

model.save('DocModel.model')
model.save_weights('DocWeights.h5')
```



After the end of 60,000 epochs, the mean squared error got reduced to 663. Thus, as per the prediction comparison, the neural network proved most efficient.

6] Converting the Test dataset as per steps 1,2 and 3.

[1] Given Testing dataset:

```
Qualification,Experience,Rating,Place,Profile,Miscellaneous_Info
MBBS,35 years experience,,"Ghatkopar East, Mumbai",General Medicine,
"MBBS, Diploma in Otorhinolaryngology (DLO)",31 years experience,,"West Marredpally, Hyderabad",ENT Specialist,
"MBBS, DDVL",40 years experience,70%,"KK Nagar, Chennai",Dermatologists,"70% 4 Feedback KK Nagar, Chennai"
BAMS,0 years experience,,"New Ashok Nagar, Delhi",Ayurveda,
"BDS, MDS - Conservative Dentistry & Endodontics",16 years experience,100%,"Kanakpura Road, Bangalore",Dentist,General Dentistry
"BDS, MDS",14 years experience,90%,"Velachery, Chennai",Dentist,Acrylic Partial Denture Impaction / Impacted Tooth Extraction BPS
"MBBS, Diploma in Otorhinolaryngology (DLO)",23 years experience,94%,"Frazer Town, Bangalore",ENT Specialist,"94% 6 Feedback Fraz
"BDS, MDS - Pedodontics",9 years experience,94%,"Attapur, Hyderabad",Dentist,RCT - Root Canal Treatment Ceramic Veneers / Crowns
"MD - Ayurveda Medicine, BAMS, Yoga Teachers Training Course, M. D. IN KAYACHIKISTA",11 years experience,99%,"Banashankari, Banga
BHMS,44 years experience,,"Mayur Vihar Ph-I, Delhi",Homeopath,
BDS,4 years experience,100%,"Erragadda, Hyderabad",Dentist,Dental Fillings Zirconia Crowns RCT - Single Sitting
"MBBS, FCD - Diabetology",34 years experience,,"Gandhipuram, Coimbatore",General Medicine,
"MBBS, DDV",11 years experience,98%,"Kemps Corner, Mumbai",Dermatologists,"98% 28 Feedback Kemps Corner, Mumbai"
"MBBS, MD - Dermatology",13 years experience,99%,"Banjara Hills, Hyderabad",Dermatologists,Laser Hair Removal - Face Skin Tag Tre
MS - ENT,4 years experience,,"Madandapuram, Chennai",ENT Specialist,
BDS,4 years experience,,"Mukherjee Nagar, Delhi",Dentist,"Mukherjee Nagar, Delhi ₹250 Visits Today"
"MBBS, MD - Dermatology",16 years experience,87%,"Tis Hazari, Delhi",Dermatologists,"87% 9 Feedback Tis Hazari, Delhi"
"MBBS, Diploma in Otorhinolaryngology (DLO), MS - ENT, DNB - ENT, MNAMS (Membership of the National Academy)",28 years experience
"BDS, MDS - Conservative Dentistry & Endodontics",6 years experience,,"Nagawara, Bangalore",Dentist,Ceramic Veneers / Crowns Cosm
"MBBS, MD - Dermatology",32 years experience,90%,"Basavanagudi, Bangalore",Dermatologists,"90% 3 Feedback Basavanagudi, Bangalore
"BHMS, Diploma in Naturopathy & Yogic Science (DNYS), Diploma in Diet and Nutrition, Diploma in Cosmetology",11 years experience,
BDS,3 years experience,,"Kattakada, Thiruvananthapuram",Dentist,
```

[2] After step 1:

```
,Qualification,Experience,Rating,Place,Profile
0,MBBS,35,0,Mumbai,General Medicine
1,"MBBS,DLO",31,0,Hyderabad,ENT Specialist
2,"MBBS,DDVL,DDV",40,70,Chennai,Dermatologists
3,BAMS,0,0,Delhi,Ayurveda
4,"MD,BDS,MDS",16,100,Bangalore,Dentist
5,"MD,BDS,MDS",14,90,Chennai,Dentist
6,"MBBS,DLO",23,94,Bangalore,ENT Specialist
7,"MD,BDS,MDS",9,94,Hyderabad,Dentist
8,"MD,BAMS",11,99,Bangalore,Ayurveda
9,BHMS,44,0,Delhi,Homeopath
10,BDS,4,100,Hyderabad,Dentist
11,MBBS,34,0,Coimbatore,General Medicine
12,"MBBS,DDV",11,98,Mumbai,Dermatologists
13,"MD,MBBS",13,99,Hyderabad,Dermatologists
14,MS,4,0,Chennai,ENT Specialist
15,BDS,4,0,Delhi,Dentist
16,"MD,MBBS",16,87,Delhi,Dermatologists
17,"MBBS,DNB,DLO",28,64,Bangalore,ENT Specialist
18,"MD,BDS,MDS",6,0,Bangalore,Dentist
19,"MD,MBBS",32,90,Bangalore,Dermatologists
20,BHMS,11,0,Mumbai,Homeopath
```

[3] After step 2:

```
,Qualification,Experiece,Ratings,Place,Profile
0,MBBS,35.0,94.44147157190635,Mumbai,General Medicine
1,MBBS,31.0,94.44147157190635,Hyderabad,ENT Specialist
2,MBBS,40.0,70.0,Chennai,Dermatologists
3,BAMS,18.1579754601227,94.44147157190635,Delhi,Ayurveda
4,MD,16.0,100.0,Bangalore,Dentist
5,MD,14.0,90.0,Chennai,Dentist
6,MBBS,23.0,94.0,Bangalore,ENT Specialist
7,MD,9.0,94.0,Hyderabad,Dentist
8,MD,11.0,99.0,Bangalore,Ayurveda
9,BHMS,44.0,94.44147157190635,Delhi,Homeopath
10,BDS,4.0,100.0,Hyderabad,Dentist
11,MBBS,34.0,94.44147157190635,Coimbatore,General Medicine
12,MBBS,11.0,98.0,Mumbai,Dermatologists
13,MD,13.0,99.0,Hyderabad,Dermatologists
14,MS,4.0,94.44147157190635,Chennai,ENT Specialist
15,BDS,4.0,94.44147157190635,Delhi,Dentist
16,MD,16.0,87.0,Delhi,Dermatologists
17,MBBS,28.0,64.0,Bangalore,ENT Specialist
18,MD,6.0,94.44147157190635,Bangalore,Dentist
19,MD,32.0,90.0,Bangalore,Dermatologists
20,BHMS,11.0,94.44147157190635,Mumbai,Homeopath
```

[4] After step 3:

```
,Qualification,Experiece,Ratings,Place,Profile
0,0.4923870859997662,1.5120038368498963,0.0,0.4923870859997667,0.49238708599976677
1,0.4923870859997662,1.1529012044389562,0.0,0.4923870859997667,0.49238708599976677
2,0.4923870859997662,1.9608821273635704,-4.8033641038777315,0.4923870859997667,0.49238708599976677
3,-0.698137542719553,0.0,0.0,-0.6981375427195524,-0.6981375427195524
4,0.7304920117436301,-0.19373366710206746,1.0923906870066744,0.7304920117436307,0.7304920117436309
5,0.7304920117436301,-0.3732849833075373,-0.8728609099547946,0.7304920117436307,0.7304920117436309
6,0.4923870859997662,0.4346959396170769,-0.08676027117020682,0.4923870859997667,0.49238708599976677
7,0.7304920117436301,-0.8221632738212118,-0.08676027117020682,0.7304920117436307,0.7304920117436309
8,0.7304920117436301,-0.642611957615742,0.8958655273105272,0.7304920117436307,0.7304920117436309
9,-1.6505572456950084,2.3199847597745102,0.0,-1.6505572456950075,-1.6505572456950077
10,-1.8886621714388723,-1.2710415643348865,1.0923906870066744,-1.8886621714388716,-1.8886621714388716
11,0.4923870859997662,1.422228178747161,0.0,0.4923870859997667,0.49238708599976677
12,0.4923870859997662,-0.642611957615742,0.699340367614381,0.4923870859997667,0.49238708599976677
13,0.7304920117436301,-0.4630606414102722,0.8958655273105272,0.7304920117436307,0.7304920117436309
14,0.9685969374874941,-1.2710415643348865,0.0,0.9685969374874941,0.9685969374874944
15,-1.8886621714388723,-1.2710415643348865,0.0,-1.8886621714388716,-1.8886621714388716
16,0.7304920117436301,-0.19373366710206746,-1.4624363890432353,0.7304920117436307,0.7304920117436309
17,0.4923870859997662,0.8835742301307514,-5.982515062054612,0.4923870859997667,0.49238708599976677
18,0.7304920117436301,-1.0914902481294166,0.0,0.7304920117436307,0.7304920117436309
```

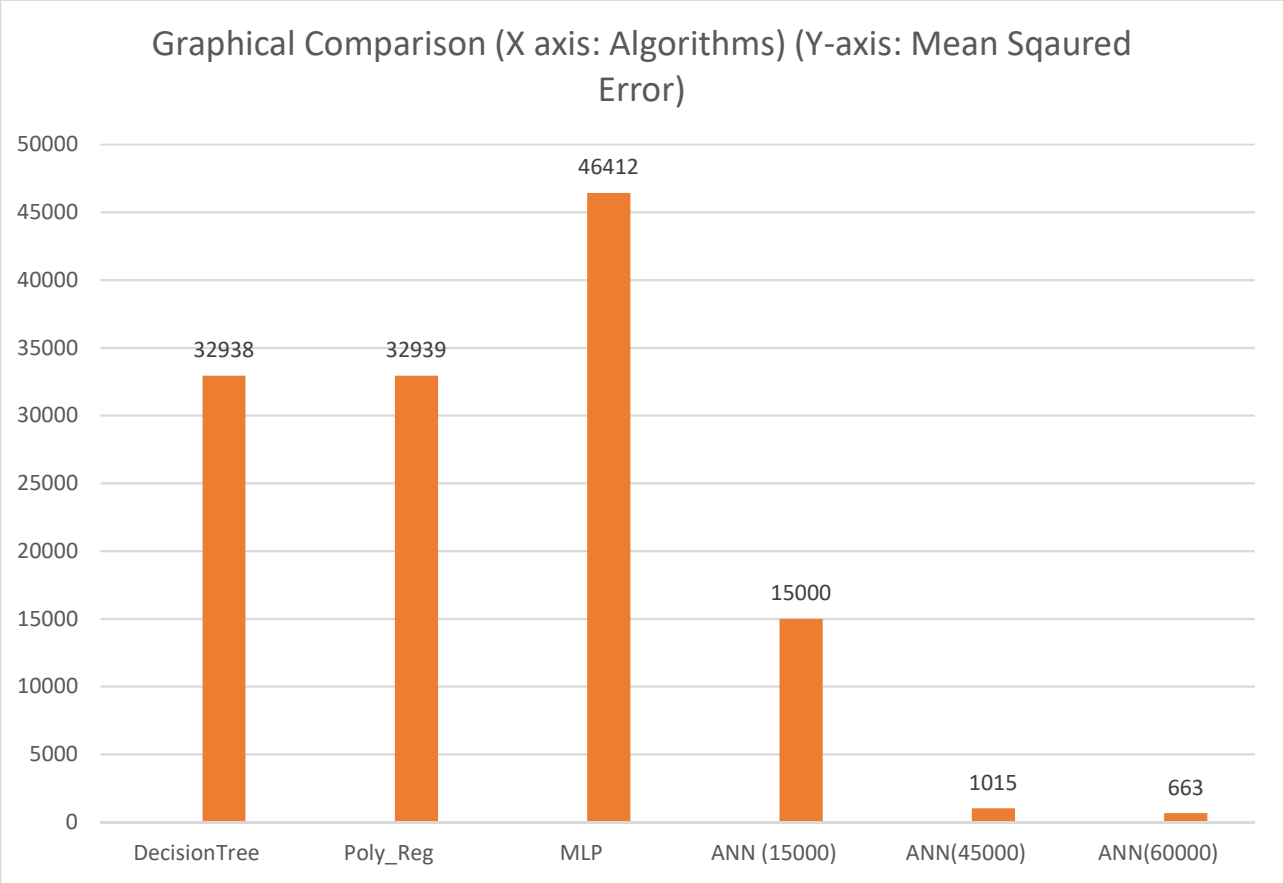
[7] Predicting values for testing dataset

```
from keras.models import load_model
import math


DATA = pd.read_csv('FinalTestS.csv')

X = DATA.iloc[:,1:].values
Y = []
model = load_model('14587.model')

for value in X:
    v = []
    for i in value:
        v.append(i)
    ans = model.predict(np.array([v]))
    Y.append(ans)



Results = []
for i in Y:
    temp=i[0][0]
    temp=round(temp)
    Results.append(int(temp))



Df = pd.DataFrame(Results)
Df.to_csv('Ans.csv',index=False)
```

Final CSV with predicted values: (To be submitted to Hackathon)

```
Fees
270
376
525
255
401
221
352
290
341
276
309
186
286
394
303
284
615
329
262
347
333
200
353
325
291
```

Comparison of prediction algorithms



Obtained conclusions from the project: -

[1] Decision Tree Regressor and polynomial regression fail to predict accurate values for real world datasets obtained from survey.

[2] The accuracy of ANN is directly proportional to the number of epochs on which it is trained.

[3] ANN proves best for prediction algorithms of real-world datasets.

[4] Data cleaning is essential for removing noise and making the dataset compatible for prediction algorithms.

[5] The accuracy of prediction highly depends on two things: - a] The cleaning process b] The prediction algorithm

[6] Generally, standardization process proves best for regressive actions over a dataset.