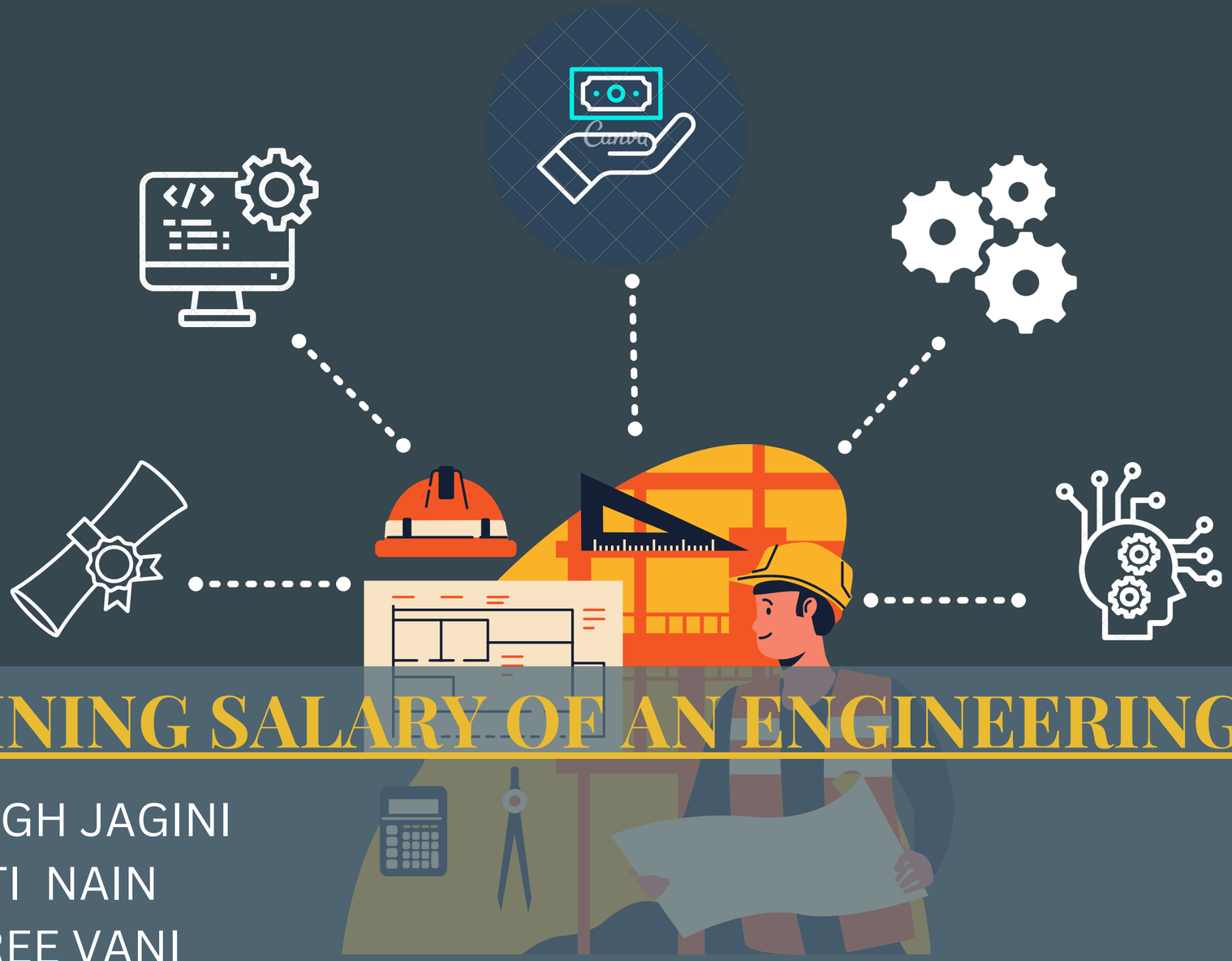


DETERMINING SALARY OF AN ENGINEERING GRADUATE

GROUP 5: AMOGH JAGINI
JYOTI NAIN
B. SREE VANI

BHAVAN'S VIVEKANADA COLLEGE,SAINIKPURI | BSC HONS. IN DATA SCIENCE

BACKGROUND

A survey was conducted to record various engineering graduate attributes such as college grades, candidate skills, the proximity of the college to industrial hubs, their specialization and market conditions for specific industries.

OBJECTIVE

On the basis of these various factors, our objective is to determine the salary of an engineering graduate in India.

THE PATH

We followed numerous ML Algorithms to fit a predictive model to this data inorder to determine Salary of an Engineering graduate in India.

ABOUT THE DATA

- **ID:** A UNIQUE ID TO IDENTIFY A CANDIDATE
- **SALARY:** ANNUAL CTC OFFERED TO THE CANDIDATE
- **GENDER:** CANDIDATE'S GENDER
- **DOB:** DATE OF BIRTH OF THE CANDIDATE
- **10PERCENTAGE:** MARKS OBTAINED IN GRADE 10 EXAMINATIONS
- **10BOARD:** THE SCHOOL BOARD WHOSE CURRICULUM THE CANDIDATE FOLLOWED IN GRADE 10
- **12GRADUATION:** YEAR OF GRADUATION - SENIOR YEAR HIGH SCHOOL
- **12PERCENTAGE:** OVERALL MARKS OBTAINED IN GRADE 12 EXAMINATIONS
- **12BOARD:** THE SCHOOL BOARD WHOSE CURRICULUM THE CANDIDATE FOLLOWED
- **COLLEGEID:** UNIQUE ID IDENTIFYING THE UNIVERSITY/COLLEGE WHICH THE CANDIDATE ATTENDED
- **COLLEGETIER:** EACH COLLEGE HAS BEEN ANNOTATED AS 1 OR 2.
- **DEGREE:** DEGREE OBTAINED/PURSUED BY THE CANDIDATE
- **SPECIALIZATION:** SPECIALIZATION PURSUED BY THE CANDIDATE
- **COLLEGEGPA:** AGGREGATE GPA AT GRADUATION
- **COLLEGECITYID:** A UNIQUE ID TO IDENTIFY THE CITY IN WHICH THE COLLEGE IS LOCATED IN.
- **COLLEGECITYTIER:** THE TIER OF THE CITY IN WHICH THE COLLEGE IS LOCATED IN.
- **COLLEGESTATE:** NAME OF THE STATE IN WHICH THE COLLEGE IS LOCATED
- **GRADUATIONYEAR:** YEAR OF GRADUATION (BACHELOR'S DEGREE)
- **AMCAT SCORES**- 16 SUBJECT-WISE COLUMNS

The data around this population of graduates had 34 attributes/features and 2998 observations.

Categorical Columns

GENDER
COLLEGE TIER
10BOARD
12BOARD
DEGREE
SPECIALIZATION
COLLEGECITYTIER
COLLEGESTATE

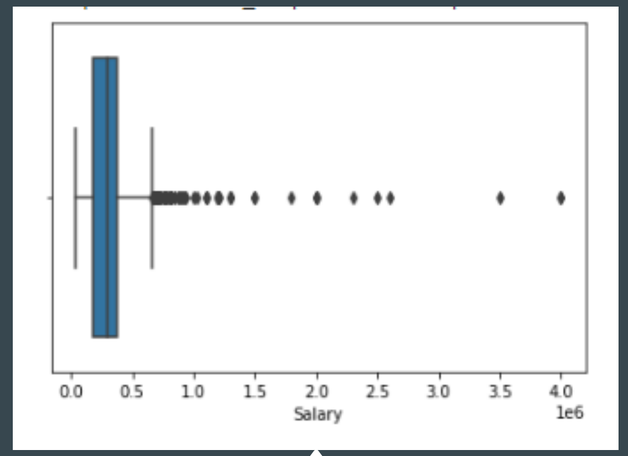
Numerical Columns

ID
DOB
10PERCENTAGE
12GRADUATION
12PERCENTAGE
COLLEGE ID
COLLEGEGPA
COLLEGECITYID
ENGLISH
LOGICAL
QUANT
DOMAIN
GRADUATIONYEAR
COMPUTERPROGRAMMING
ELECTRONICSANDSEMICON
COMPUTERSCIENCE
MECHANICALENGG
ELECTRICALENGG
TELECOMENGG
CIVILENGG
CONSCIENTIOUSNESS
AGREEABLENESS
EXTRAVERSION
NUEROTICISM
OPENESS_TO_EXPERIENCE
SALARY

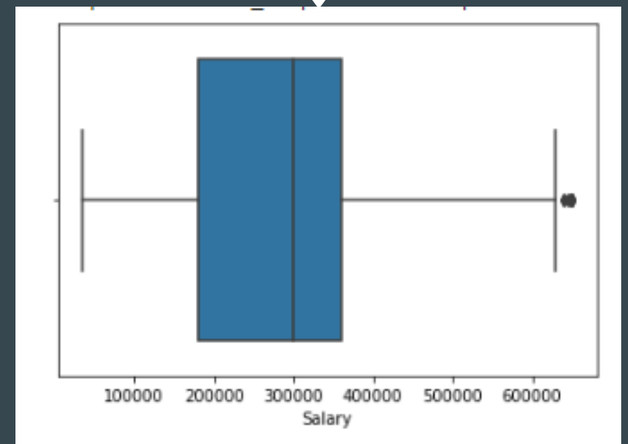
DATA AND DATA QUALITY CHECK

- There are no null values present.
- Replacing -1 with 0 in AMCAT score columns.
- Replacing male with 0 and female with 1

```
Gender                      0
DOB                         0
10percentage                0
10board                     0
12graduation                0
12percentage                0
12board                     0
CollegeID                   0
CollegeTier                 0
Degree                      0
Specialization              0
collegeGPA                  0
CollegeCityID               0
CollegeCityTier             0
CollegeState                0
GraduationYear              0
English                     0
Logical                     0
Quant                       0
Domain                      0
ComputerProgramming         0
ElectronicsAndSemicon       0
ComputerScience             0
MechanicalEngg              0
ElectricalEngg              0
TelecomEngg                 0
CivilEngg                   0
conscientiousness           0
agreeableness               0
extraversion                0
nueroticism                 0
openess_to_experience       0
Salary                      0
dtype: int64
```



Removing the outliers using inter-quartile range

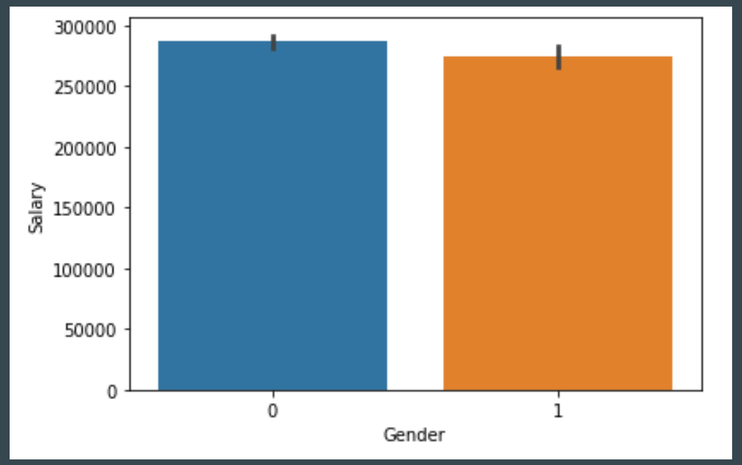


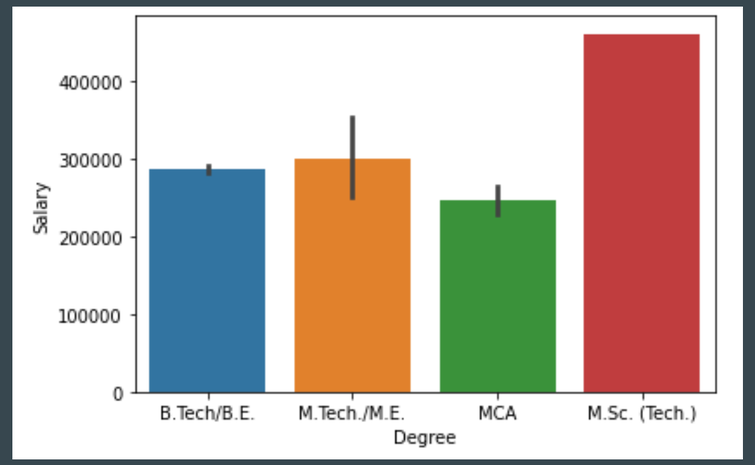
EXPLORATORY DATA ANALYSIS

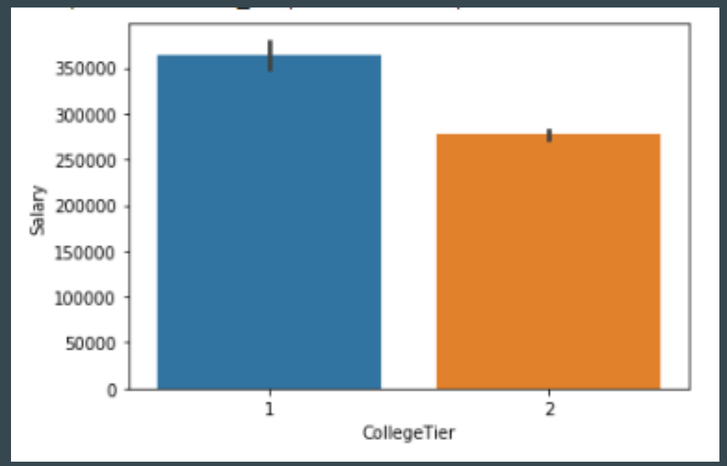
BAR PLOT

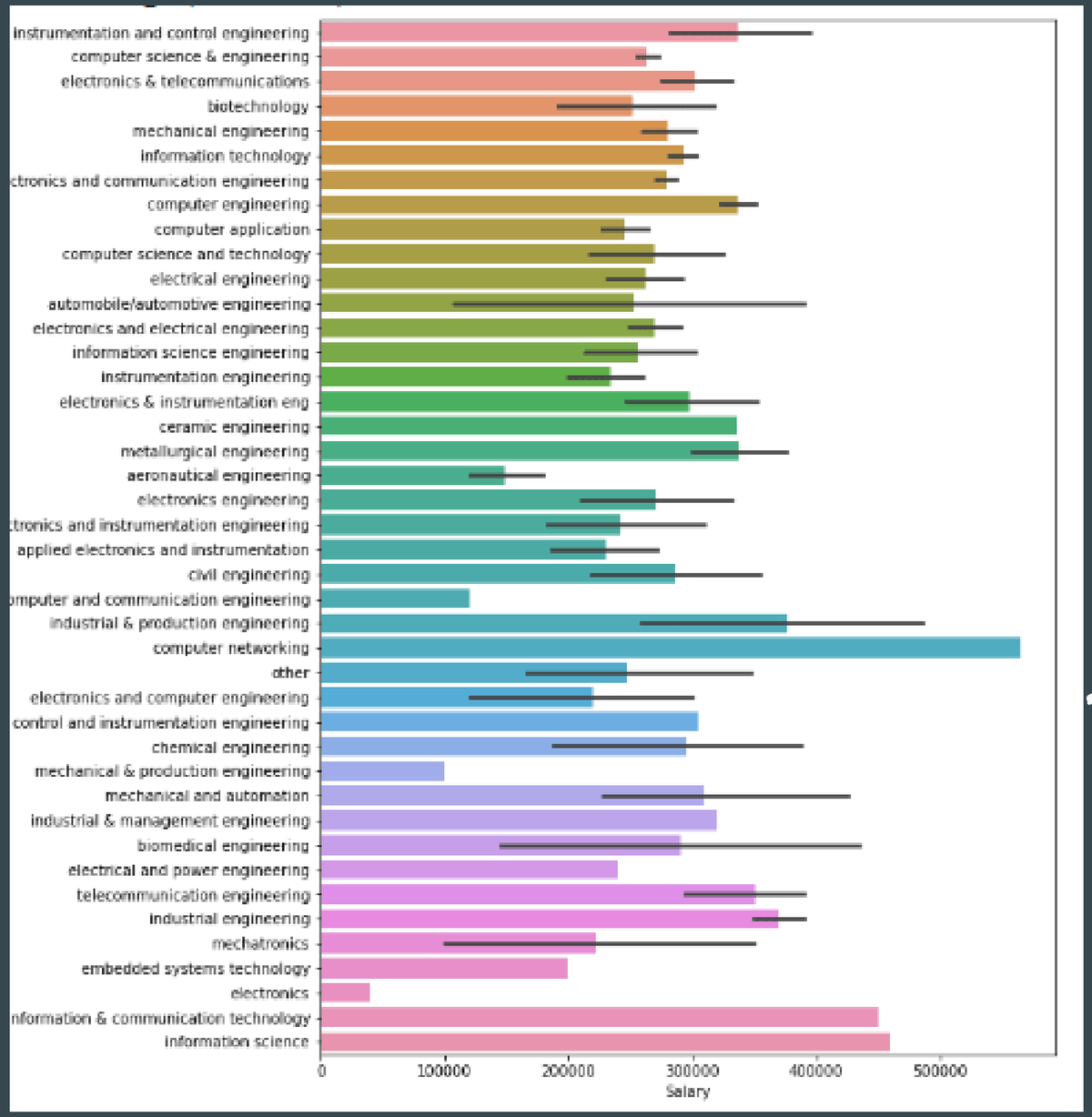
- These histograms show the salary based on several different variables of the data set.

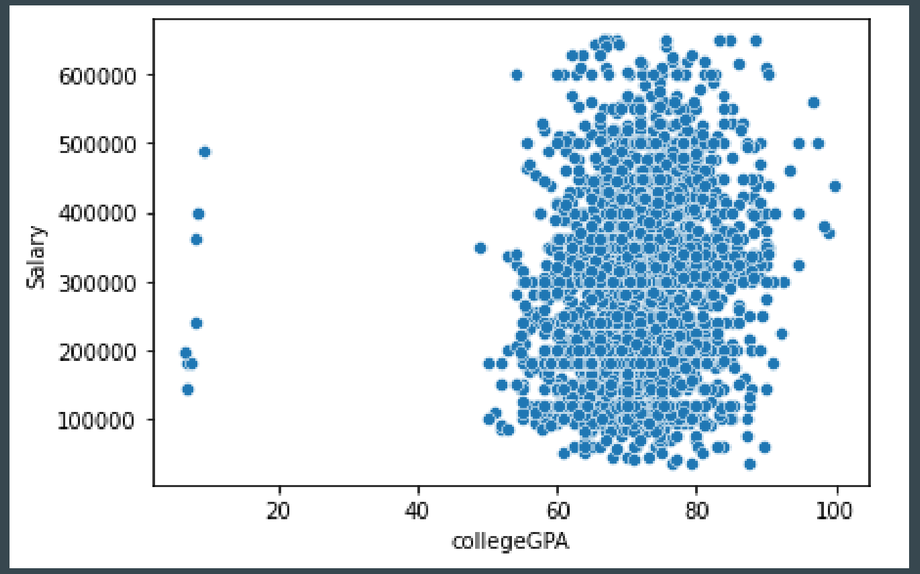
- Some of these variables are gender, degree and collegeTier.







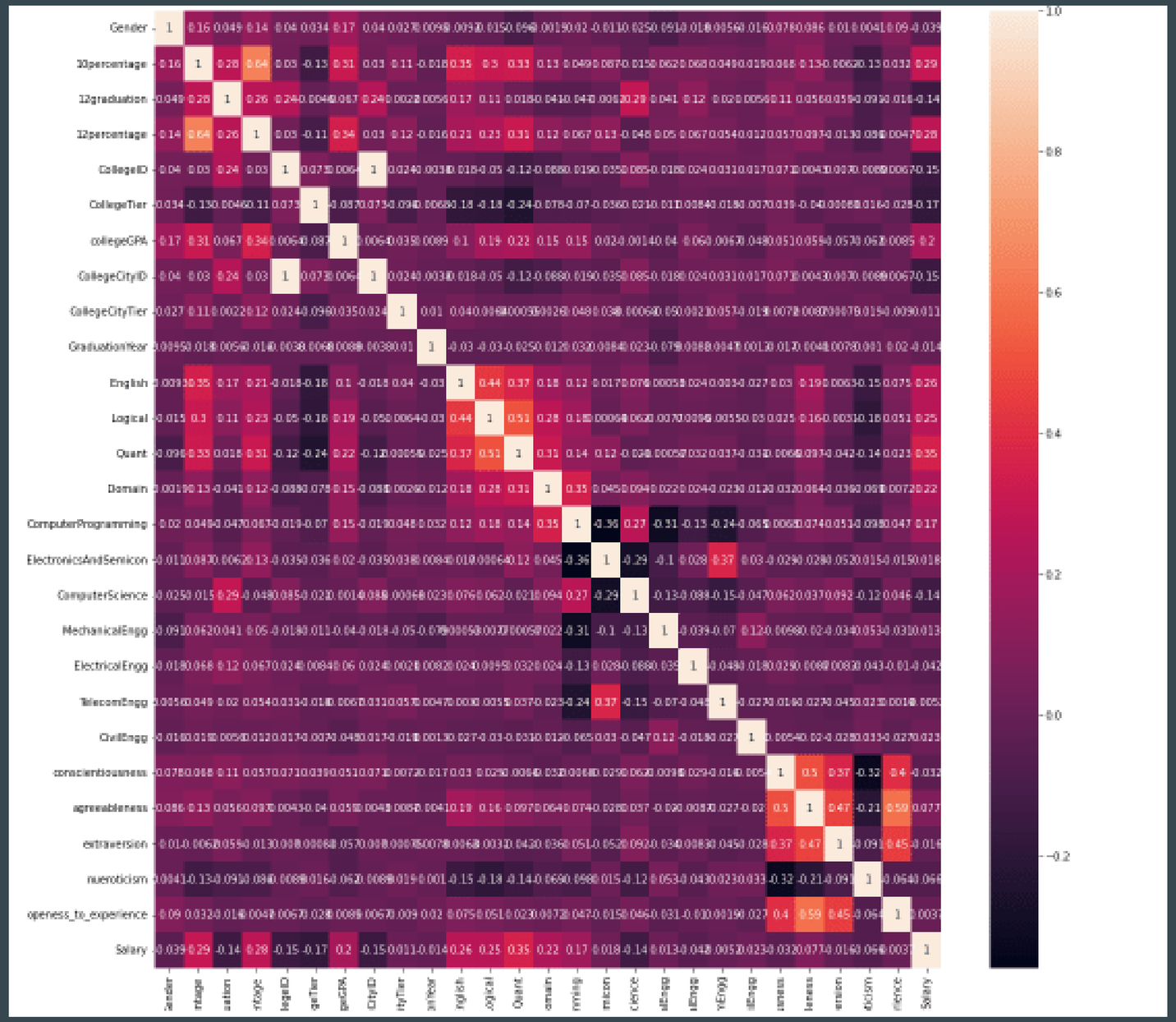


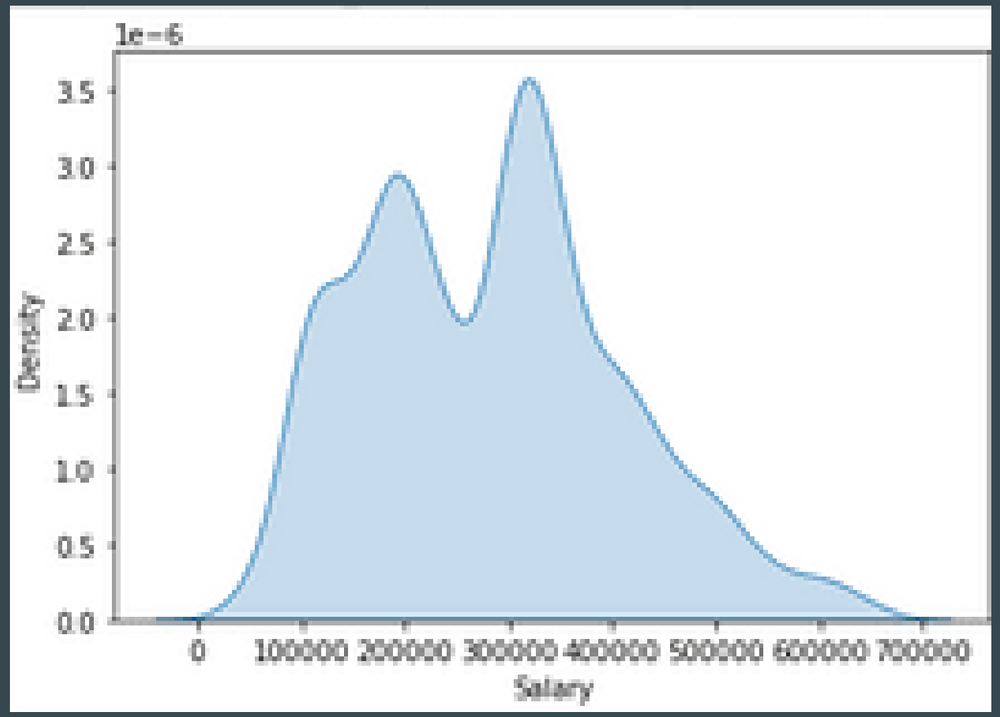


Salary V/S College GPA

Here computer networking graduate have the highest average salary whereas electronics graduates have the lowest.

EXPLORATORY DATA ANALYSIS

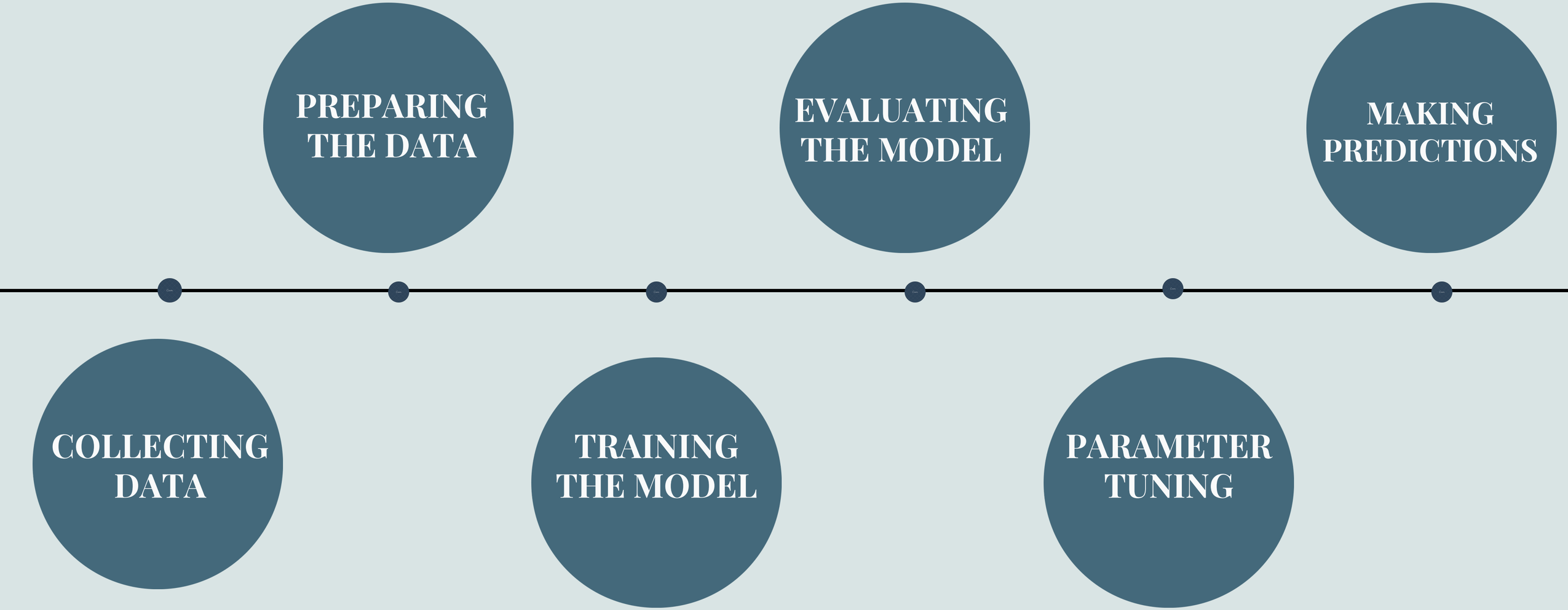




EXPLORATORY DATA ANALYSIS INSIGHTS

- We dropped 8 columns : DOB,COLLEGE ID, COLLEGE CITY ID, 10BOARD etc.
- We dummified two columns: degree and specialization.
- Computer networking graduates are having the highest salary.
- M.Sc.(Tech) graduates are earning the highest salary.
- Graduates who scored better in AMCAT are having more average salary.
- We removed outliers using inter-quartile range

Steps involved in Analysis of Machine Learning Algorithm



LINEAR REGRESSION

METRIC USED:
Mean Absolute Error (MAE)

80 - 20 is giving the least MAE value compared to other train-test splits.

TRAIN-TEST SPLIT	MAE VALUES
90-10	88040.126
80-20	87295.008
75-25	87876.935
70-30	89075.589
60-40	89705.106

NEURAL NETWORK

ADAM

Train-Test Proportion	Architecture	Optimizer	Epochs	MAE
70--30	30--20--10--5--1	Adam	50	93139.8438
			100	91903.4766
			200	90205.3359
			300	89396.5469
			400	89133.6250
			500	88910.3438
70--30	40--30--20--10--5--1	Adam	50	92457.0625
			100	91162.4766
			200	89709.1484
			300	89092.8203
			400	89025.5469
			500	88748.1172
70--30	50--40--30--20--10--5--1	Adam	50	91926.3828
			100	90279.3516
			200	89908.6406
			300	89764.5391
			400	88858.8125
			500	88497.4219
70--30	60--50--40--30--20--10--5--1	Adam	50	91525.4062
			100	90136.2656
			200	90287.5625
			300	90194.8281
			400	89104.6797
			500	88379.8438
70--30	50--40--30--20--10--1	Adam	50	92321.4688
			100	90493.2266
			200	90155.6797
			300	89293.5078
			400	88840.7031
			500	88503.9453

Train-Test Proportion	Architecture	Optimizer	Epochs	MAE
80--20	30--20--10--5--1	Adam	50	91496.8047
			100	90301.2344
			200	88125.0312
			300	87684.5938
			400	88490.0547
			500	86793.5073
80--20	40--30--20--10--5--1	Adam	50	90912.1016
			100	89347.6719
			200	87079.2500
			300	86956.2266
			400	88511.0781
			500	85926.3828
80--20	50--40--30--20--10--5--1	Adam	50	90585.4688
			100	89247.7656
			200	86873.6719
			300	86600.9609
			400	88688.9688
			500	86072.3906
80--20	40--20--10--1	Adam	50	91718.5000
			100	90589.1762
			200	88663.8359
			300	87721.6797
			400	87855.8203
			500	86999.1406
80--20	50--40--30--20--10--1	Adam	50	90473.5391
			100	89022.0469
			200	86972.7266
			300	86847.5547
			400	88700.3359
			500	86048.4219

Architecture : 50-40-30-20-10-5-1 under ADAM Optimizer with 500 Epochs gives us the lowest MAE value of 86072.3906

SGD

SGD is not compatible with this dataset

Train Test Proportion	Architecture	Optimizer	Epochs	MAE
70–30	10–5–1	SGD	50	Nan
			100	Nan
			200	Nan
			300	Nan
			400	Nan
			500	Nan
70–30	20–10–5–1	SGD	50	Nan
			100	Nan
			200	Nan
			300	Nan
			400	Nan
			500	Nan
70–30	30–20–10–5–1	SGD	50	Nan
			100	Nan
			200	Nan
			300	Nan
			400	Nan
			500	Nan
70–30	40–30–20–10–5–1	SGD	50	Nan
			100	Nan
			200	Nan
			300	Nan
			400	Nan
			500	Nan
70–30	50–40–30–20–10–5–1	SGD	50	Nan
			100	Nan
			200	Nan
			300	Nan
			400	Nan
			500	Nan
70–30	60–50–40–30–20–10–5–1	SGD	50	Nan
			100	Nan
			200	Nan
			300	Nan
			400	Nan
			500	Nan
70–30	40–20–1	SGD	50	Nan
			100	Nan
			200	Nan
			300	Nan
			400	Nan
			500	Nan
70–30	40–20–10–1	SGD	50	Nan
			100	Nan
			200	Nan
			300	Nan
			400	Nan
			500	Nan
70–30	50–40–20–10–1	SGD	50	Nan
			100	Nan
			200	Nan
			300	Nan
			400	Nan
			500	Nan

Train Test Proportion	Architecture	Optimizer	Epochs	MAE
80–20	10–5–1	SGD	50	Nan
			100	Nan
			200	Nan
			300	Nan
			400	Nan
			500	Nan
80–20	20–10–5–1	SGD	50	Nan
			100	Nan
			200	Nan
			300	Nan
			400	Nan
			500	Nan
80–20	30–20–10–5–1	SGD	50	Nan
			100	Nan
			200	Nan
			300	Nan
			400	Nan
			500	Nan
80–20	40–30–20–10–5–1	SGD	50	Nan
			100	Nan
			200	Nan
			300	Nan
			400	Nan
			500	Nan
80–20	50–40–30–20–10–5–1	SGD	50	Nan
			100	Nan
			200	Nan
			300	Nan
			400	Nan
			500	Nan
80–20	60–50–40–30–20–10–5–1	SGD	50	Nan
			100	Nan
			200	Nan
			300	Nan
			400	Nan
			500	Nan
80–20	40–20–1	SGD	50	Nan
			100	Nan
			200	Nan
			300	Nan
			400	Nan
			500	Nan
80–20	40–20–10–1	SGD	50	Nan
			100	Nan
			200	Nan
			300	Nan
			400	Nan
			500	Nan
80–20	50–40–20–10–1	SGD	50	Nan
			100	Nan
			200	Nan
			300	Nan
			400	Nan
			500	Nan

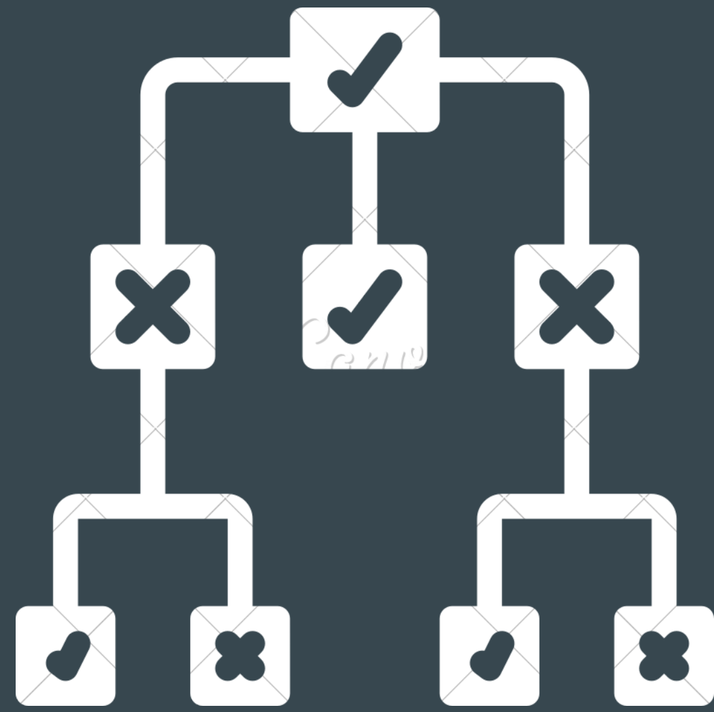
BAGGING

BAGGING REGRESSOR → 84874.041

DECISION TREE → 93247.659

RANDOM FOREST → 85234.246

Train-Test Proportions	Bagging Regressor	Decision Tree	Random Forest
80--20	86866.26710	94539.3369	86795.1369
70--30	88789.34931	93358.0268	89056.1757
90--10	84874.04109	93247.6592	85234.2465
60--40	88854.50728	93596.6151	88598.8431
75--25	87879.00000	93399.393	87550.3835

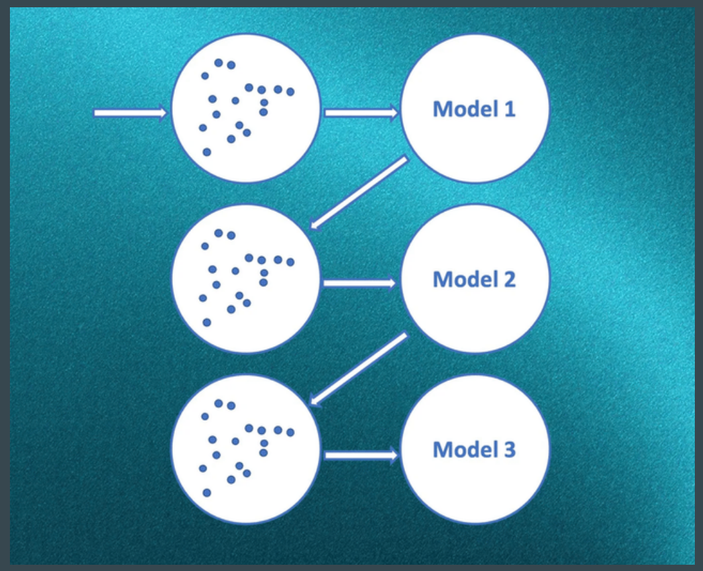


BOOSTING

ADABOOST → 88188.915

GRADIENT BOOST → 83447.868

XGBOOST → 83556.742



Train-Test Proportions	AdaBoost	GBM	XGBoost
80--20	88843.81922	86547.62905	85009.37766
70--30	90979.76535	88096.09164	86570.94945
90--10	88188.91582	83447.86811	83556.74272
60--40	91710.52945	88152.23166	86846.79318
75--25	89651.41289	87493.90547	86229.34900

SUMMARY

ALGORITHM	MAE VALUES
LINEAR REGRESSION	87295.00884
NEURAL NETWORK	86072.3906
BAGGING	84874.041
GRADIENT BOOST	**83447.86811**
DECISION TREES	93247.659
RANDOM FOREST	85234.246

CONCLUSION

Gradient boost is giving the least MAE value.

The least MAE we observed was 83447.868

Degree, specialization and AMCAT score are 3 factors which influence "salary" the most.

80-20 and 90-10 train-test splits worked best for our dataset.

HTTPS://GITHUB.COM/SREEVANI23/UNP-PROJECT/BLOB/MAIN/UNP_PROJECT.IPYNB

THANK YOU

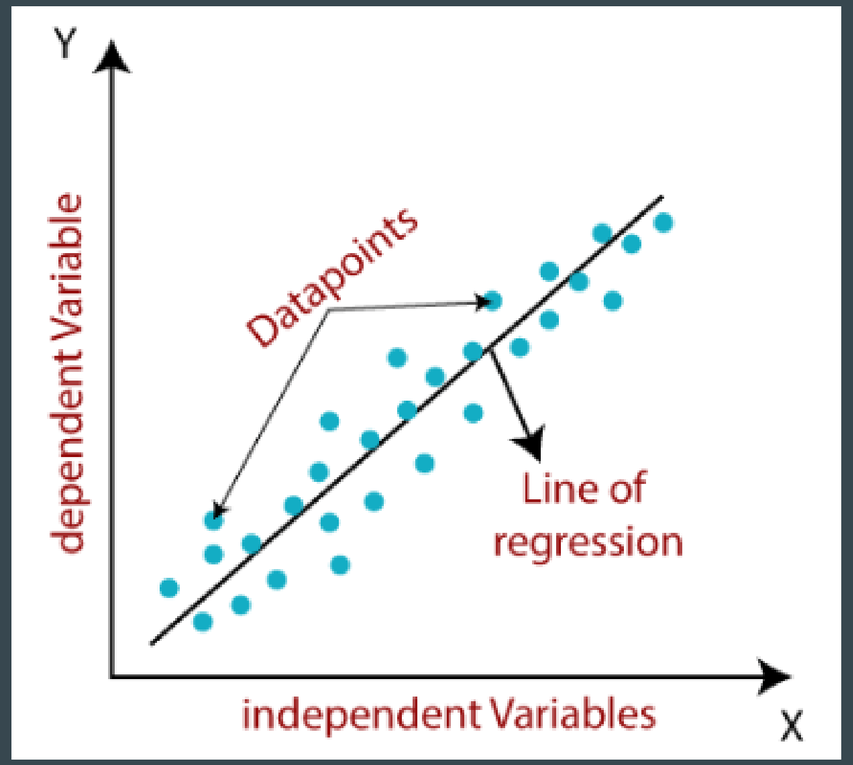
BY: AMOGH JAGINI
JYOTI NAIN
SREEVANI

APPENDIX

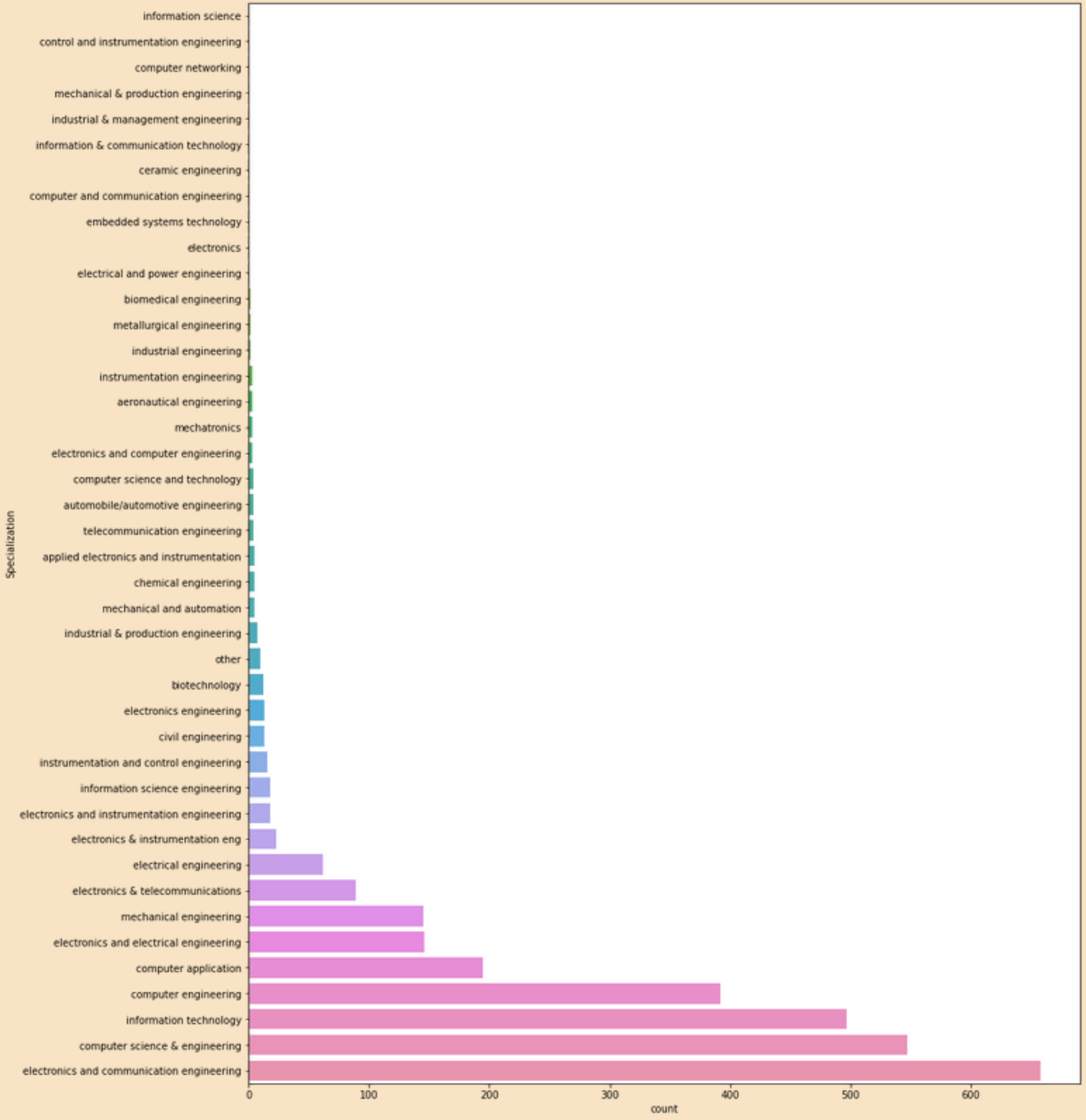
LINEAR REGRESSION

LINEAR REGRESSION ALGORITHM SHOWS A LINEAR RELATIONSHIP BETWEEN A DEPENDENT (Y) AND ONE OR MORE INDEPENDENT (Y) VARIABLES, HENCE CALLED AS LINEAR REGRESSION.

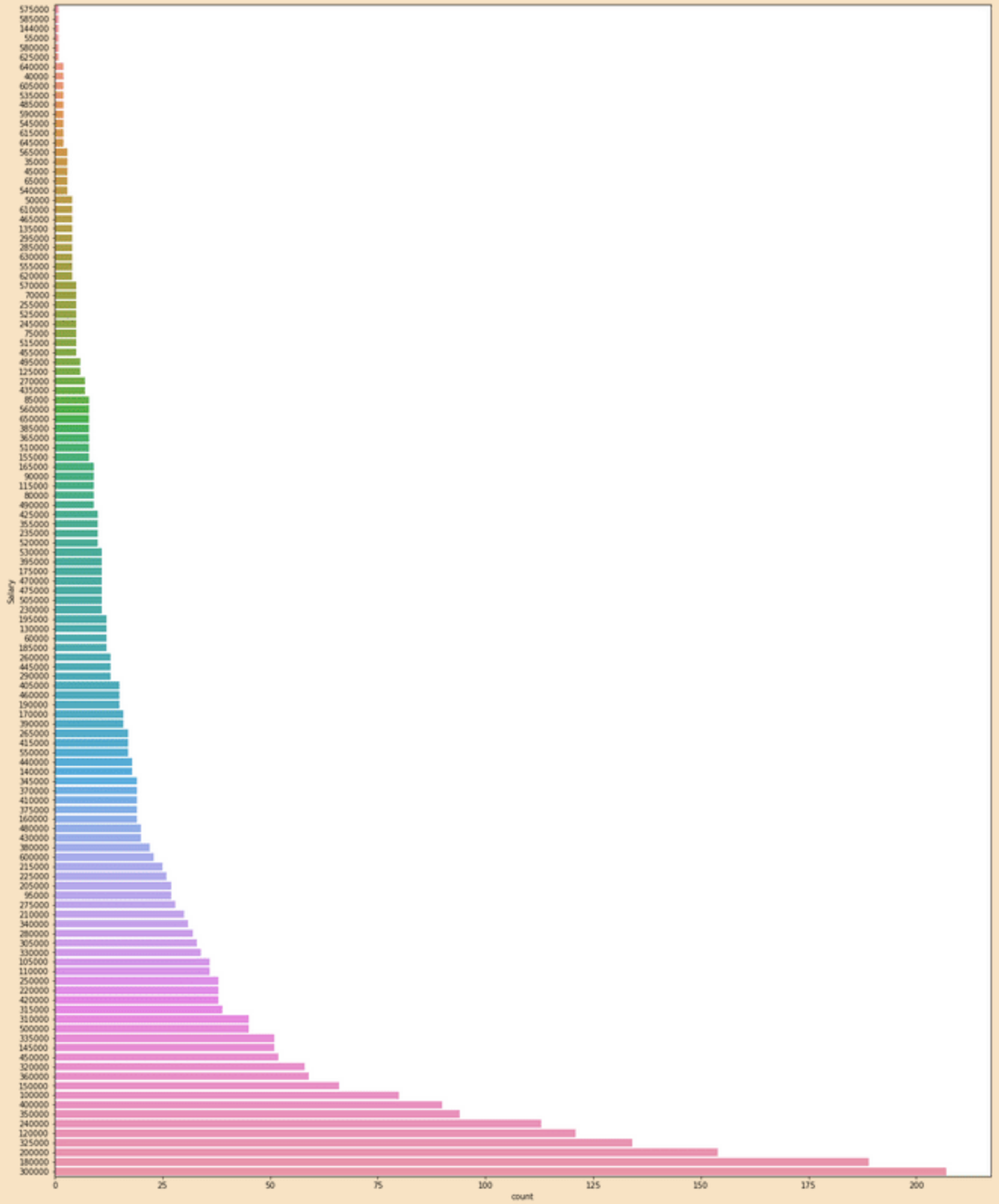
THE LINEAR REGRESSION MODEL PROVIDES A SLOPED STRAIGHT LINE REPRESENTING THE RELATIONSHIP BETWEEN THE VARIABLES.



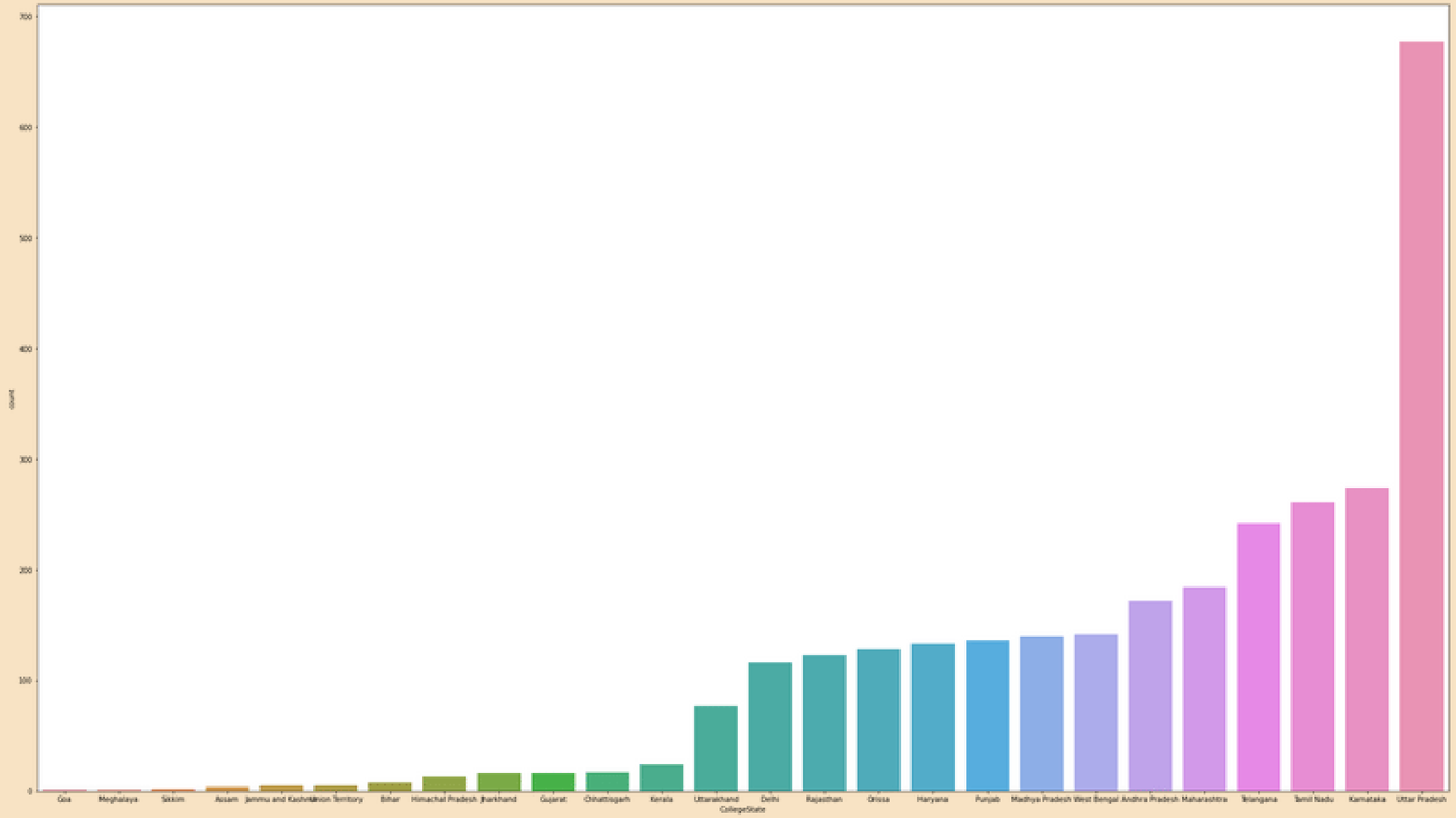
EDA



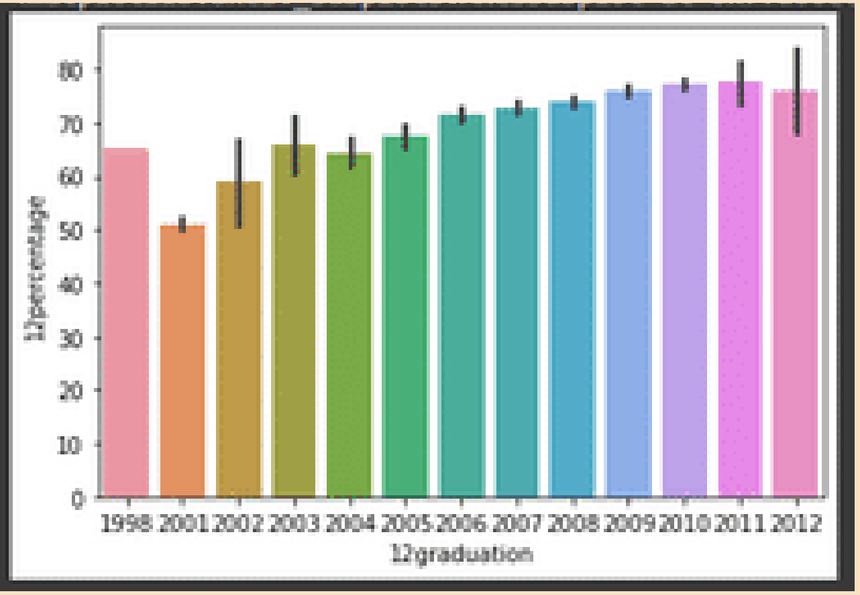
Count plot for Specialization

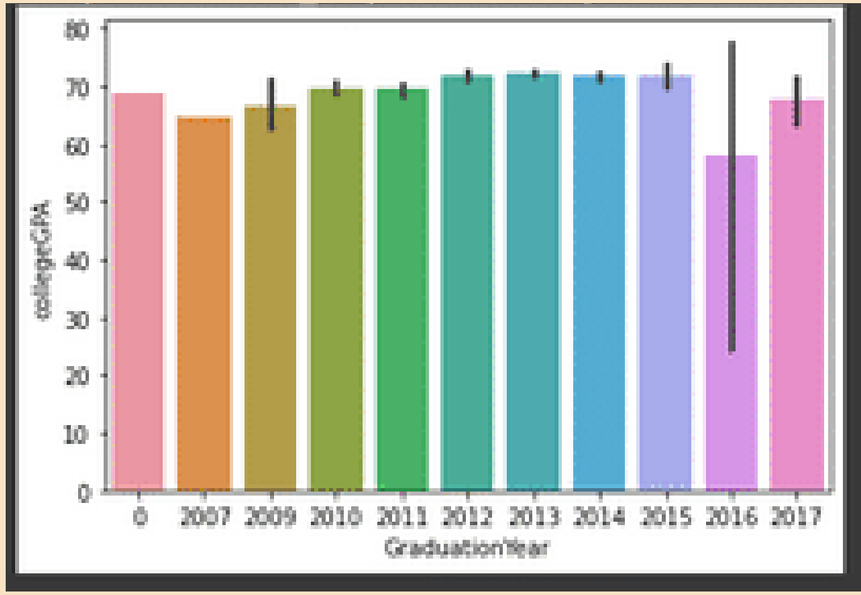


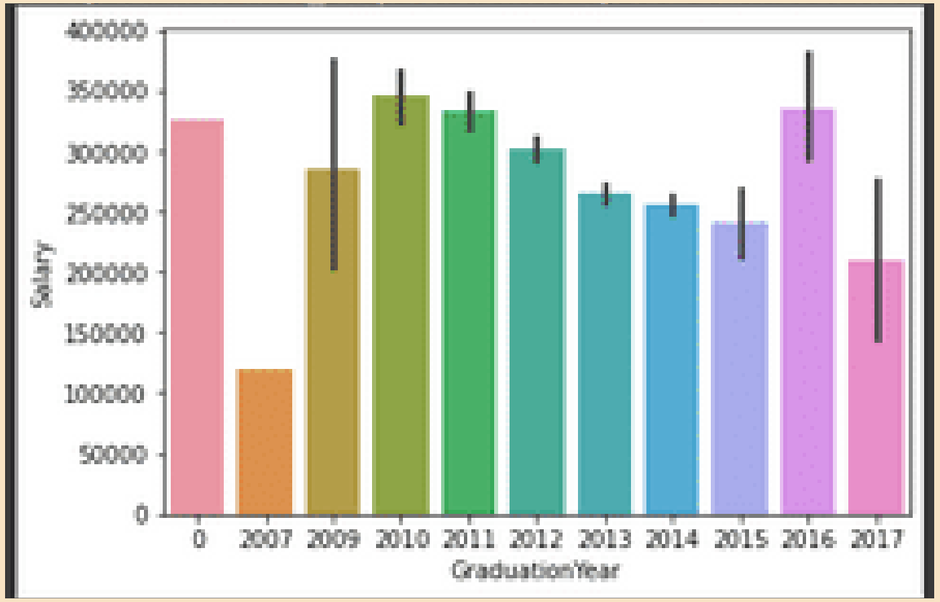
Count plot for Salary

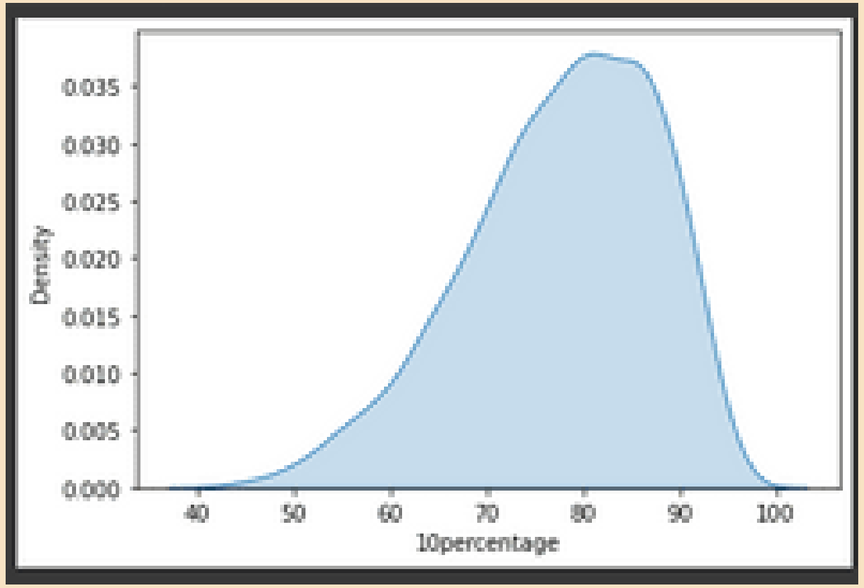


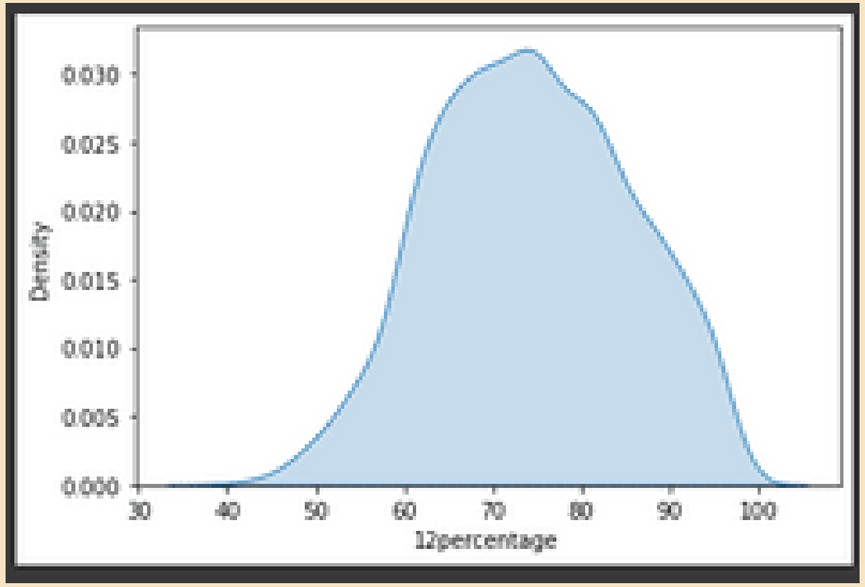
Count plot for College State

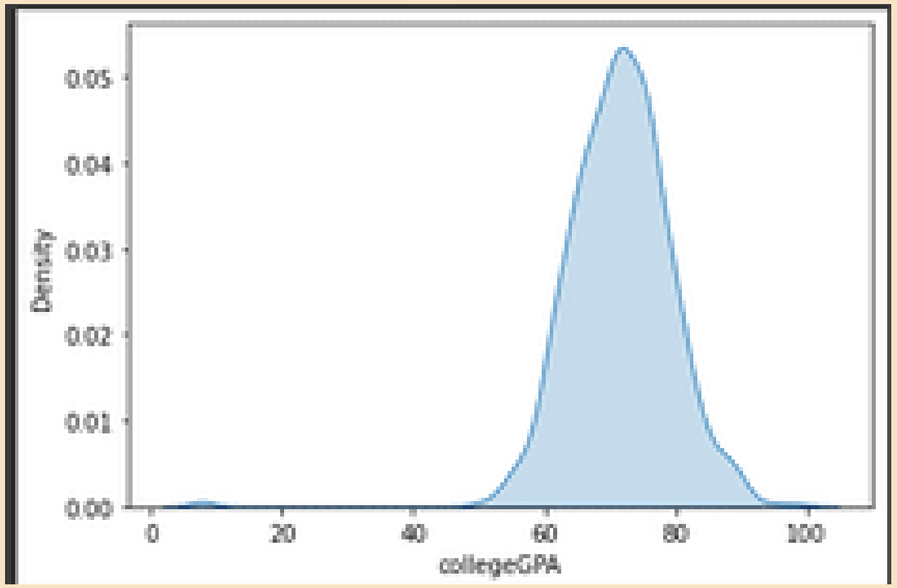

12percentage
0
10
20
30
40
50
60
70
80
1998 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012
12graduation

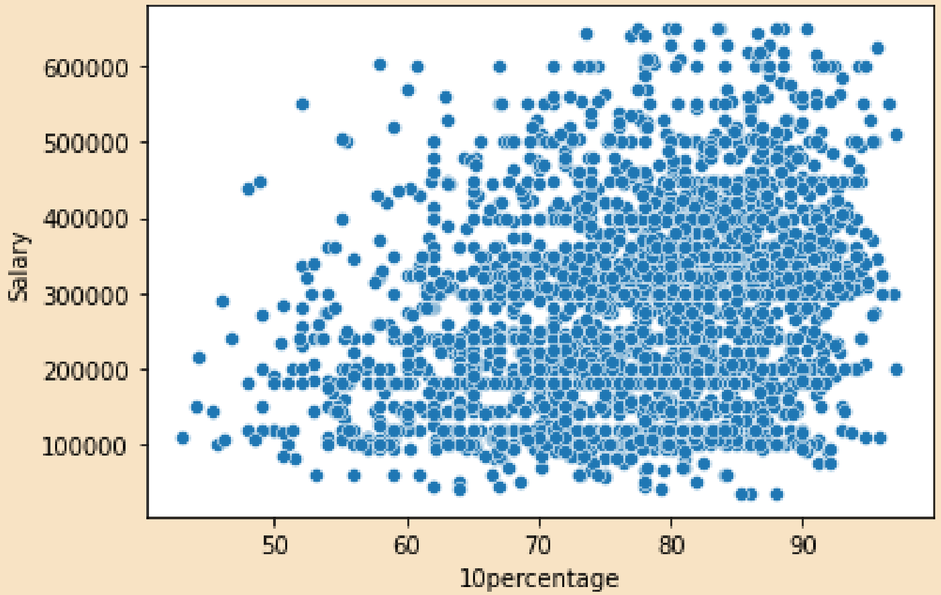

collegeGPA
0
10
20
30
40
50
60
70
80
0 2007 2009 2010 2011 2012 2013 2014 2015 2016 2017
GraduationYear

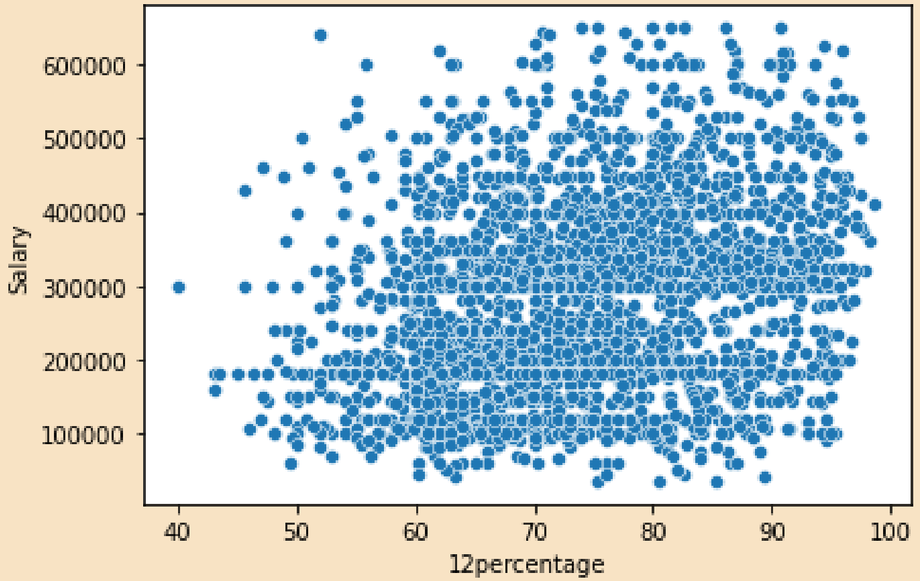

Salary
0
50000
100000
150000
200000
250000
300000
350000
400000
0 2007 2009 2010 2011 2012 2013 2014 2015 2016 2017
GraduationYear

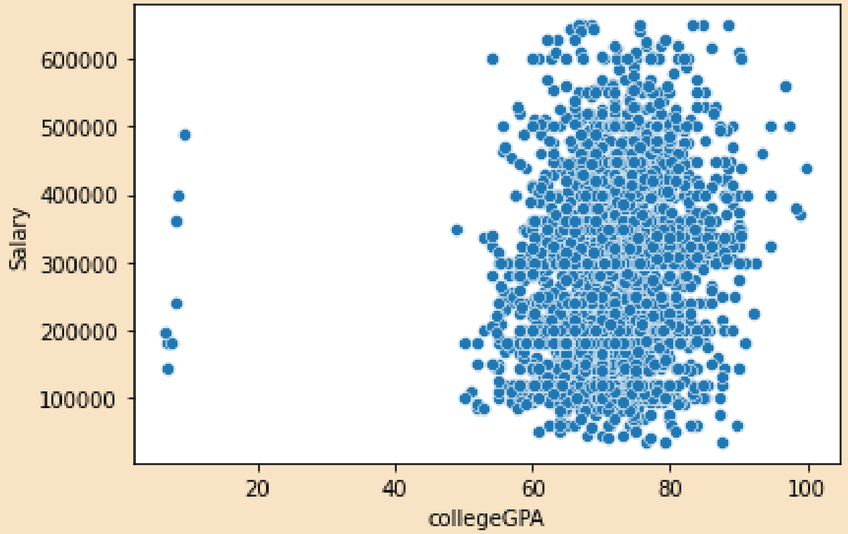

Density
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
40
50
60
70
80
90
100
10percentage

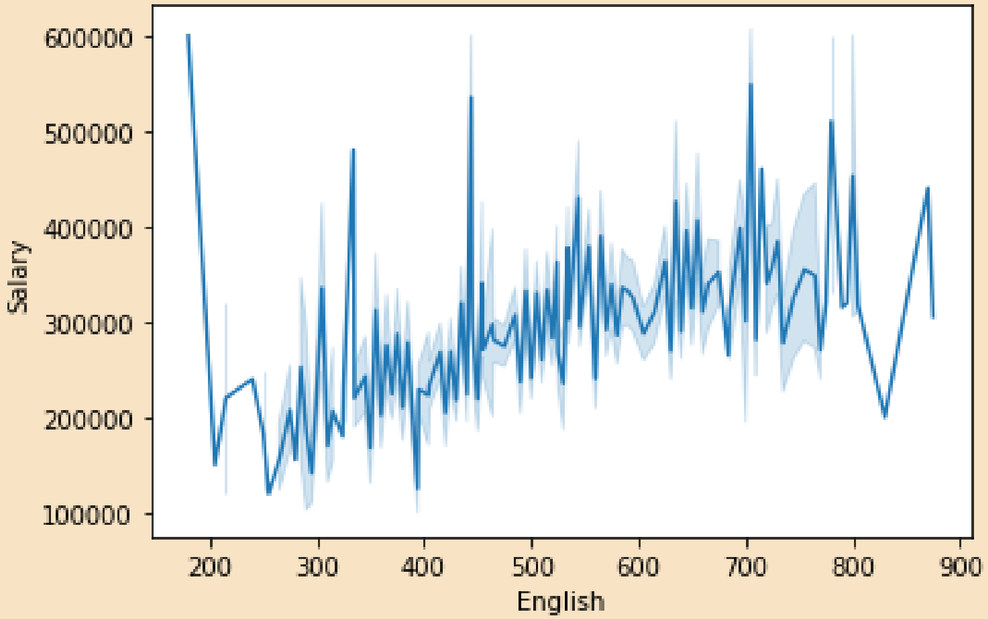

Density
0.030
0.025
0.020
0.015
0.010
0.005
0.000
30
40
50
60
70
80
90
100
12percentage

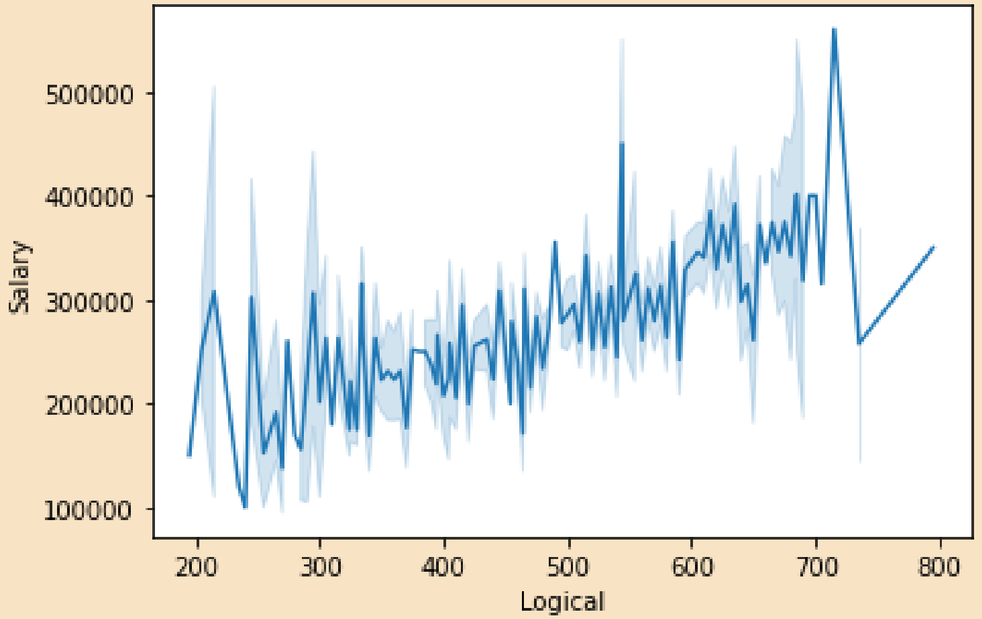

Density
0.05
0.04
0.03
0.02
0.01
0.00
0
20
40
60
80
100
collegeGPA

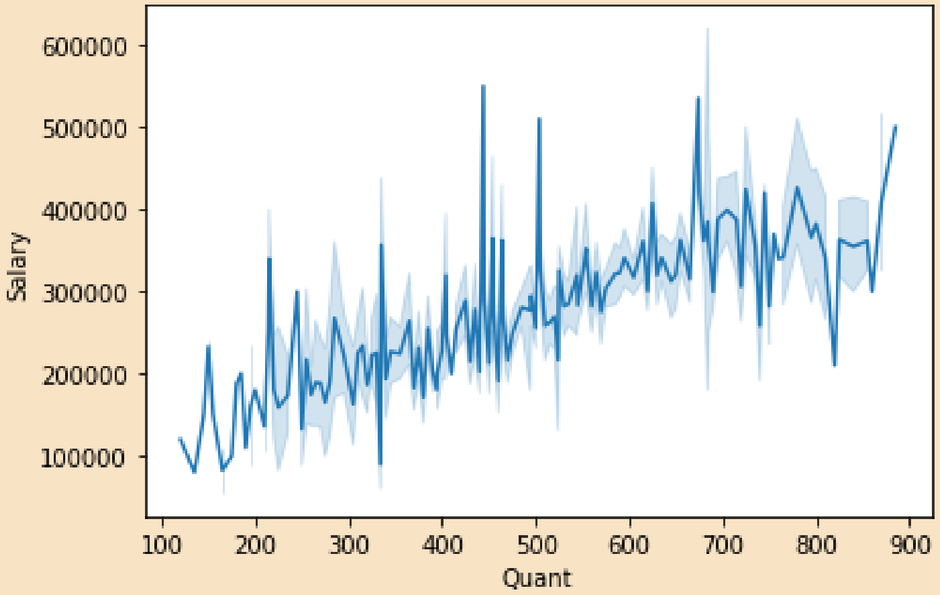

Salary
600000
500000
400000
300000
200000
100000
50
60
70
80
90
10percentage


Salary
600000
500000
400000
300000
200000
100000
40
50
60
70
80
90
100
12percentage


Salary
600000
500000
400000
300000
200000
100000
20
40
60
80
100
collegeGPA


Salary
600000
500000
400000
300000
200000
100000
200
300
400
500
600
700
800
900
English


Salary
500000
400000
300000
200000
100000
200
300
400
500
600
700
800
Logical


Salary
600000
500000
400000
300000
200000
100000
100
200
300
400
500
600
700
800
900
Quant

Dropping few columns and dummifying few columns

```
[ ] data.drop(['CollegeID', '10board','12graduation','12board' ,'CollegeCityID','CollegeState', 'CollegeCityTier','DOB'], axis = 1, inplace = True)

[ ] data = pd.get_dummies(data, columns=["Degree"])

[ ] data = pd.get_dummies(data, columns=["Specialization"])
```

Intercept

```
[ ] lm = smf.ols(formula='Salary ~ English+Logical+Quant+Domain+ElectronicsAndSemicon+ComputerScience+ElectricalEngg+TelecomEngg+conscientiousness+agreeableness+extr
    lm.params

Intercept               32292.131778
English                   164.260956
Logical                    54.719183
Quant                     253.756115
Domain                  53898.963712
ElectronicsAndSemicon     -50.922298
ComputerScience          -131.548069
ElectricalEngg           -104.226655
TelecomEngg               -21.445066
conscientiousness       -5680.100971
agreeableness            5471.412020
extraversion             2218.005698
nueroticism             -3194.104614
openess_to_experience   -2923.297539
dtype: float64
```


No of columns after dropping and dummifying

```
[ ] data.shape

    (2998, 69)

[ ] data.isnull().sum()

Gender                                  0
10percentage                            0
12percentage                            0
CollegeTier                             0
collegeGPA                              0
                                       ..
Specialization_mechanical engineering   0
Specialization_mechatronics             0
Specialization_metallurgical engineering 0
Specialization_other                    0
Specialization_telecommunication engineering 0
Length: 69, dtype: int64
```

FITTING LINEAR REGRESSION

```
[ ]  X = data.drop('Salary',axis=1)
     y = data['Salary']
```

MODEL-1

```
[ ]  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=50)
```

```
[ ]  lm1=LinearRegression()
     lm1.fit(X_train,y_train)

     LinearRegression()
```

```
[ ]  y_pred = lm1.predict(X_test)
     print(np.sqrt(mean_squared_error(y_test,y_pred)))
     print(r2_score(y_test,y_pred))

     108527.01459024395
     0.28010848888330375
```

```
[ ]  mae=metrics.mean_absolute_error(y_test,y_pred)
     mae

     87295.00884421119
```

MODEL-2

```
[ ]  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=50)

[ ]  lm2=LinearRegression()
     lm2.fit(X_train,y_train)

     LinearRegression()

[ ]  y_pred = lm2.predict(X_test)
     print(np.sqrt(mean_squared_error(y_test,y_pred)))
     print(r2_score(y_test,y_pred))

     110976.1758412874
     0.25440393031664554

[ ]  mae=metrics.mean_absolute_error(y_test,y_pred)
     mae

     89075.58990707282
```

MODEL-3

```
[ ]  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=50)

[ ]  lm3=LinearRegression()
     lm3.fit(X_train,y_train)

     LinearRegression()

[ ]  y_pred = lm3.predict(X_test)
     print(np.sqrt(mean_squared_error(y_test,y_pred)))
     print(r2_score(y_test,y_pred))

     110181.31159021903
     0.2718319212237401

[ ]  mae=metrics.mean_absolute_error(y_test,y_pred)
     mae

     88040.12643388746
```

MODEL-4

```
[ ] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=50)

[ ] lm4=LinearRegression()
    lm4.fit(X_train,y_train)

    LinearRegression()

[ ] y_pred = lm4.predict(X_test)
    print(np.sqrt(mean_squared_error(y_test,y_pred)))
    print(r2_score(y_test,y_pred))

    112270.31990916353
    0.2488258318128426 8

[ ] mae=metrics.mean_absolute_error(y_test,y_pred)
    mae

    89705.10610660515
```

MODEL-5

```
[ ] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=50)

[ ] lm5=LinearRegression()
    lm5.fit(X_train,y_train)

    LinearRegression()

[ ] y_pred = lm5.predict(X_test)
    print(np.sqrt(mean_squared_error(y_test,y_pred)))
    print(r2_score(y_test,y_pred))

    109417.87713229861
    0.26596206325639016

[ ] mae=metrics.mean_absolute_error(y_test,y_pred)
    mae

    87876.93592396312
```

K-FOLD CROSS VALIDATION FOR LINEAR REGRESSION

```
from sklearn.model_selection import cross_val_score,KFold,ShuffleSplit

linreg=LinearRegression()
cv = KFold(n_splits=5, random_state=0, shuffle=True)
```

```
[ ] scores = cross_val_score(linreg, X, y, scoring='neg_mean_absolute_error')
```

```
[ ] from numpy import mean
    from numpy import absolute

    mean(absolute(scores))
```

```
101395.40848217276
```

NEURAL NETWORK

ADAM

```
[74] import tensorflow as tf
     tf.random.set_seed(50)
     model= tf.keras.Sequential([
                                 tf.keras.layers.Dense(50),
                                 tf.keras.layers.Dense(40),
                                 tf.keras.layers.Dense(30),
                                 tf.keras.layers.Dense(20),
                                 tf.keras.layers.Dense(10),
                                 tf.keras.layers.Dense(5),
                                 tf.keras.layers.Dense(1)
     ])
```

```
[75] model.compile(loss= tf.keras.losses.mae,
                   optimizer= tf.keras.optimizers.Adam(),
                   metrics= ["mae"])
```

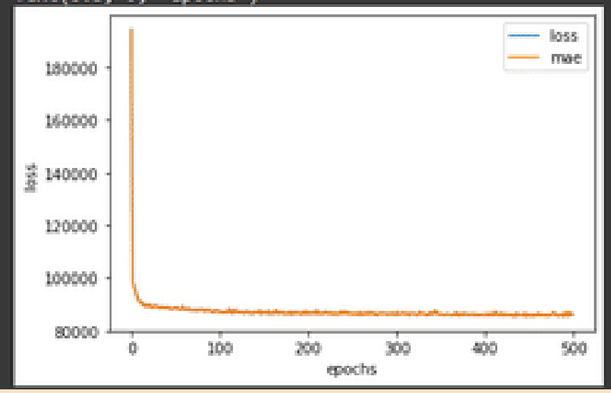
```
[76] history= model.fit(X_train, y_train, epochs= 500, verbose=0)
```

```
[77] model.evaluate(X_test, y_test)

     19/19 [==============================] - 0s 1ms/step - loss: 86072.3906 - mae: 86072.3906
     [86072.390625, 86072.390625]
```

```
[78] pd.DataFrame(history.history).plot()
     plt.ylabel("loss")
     plt.xlabel("epochs")

     Text(0.5, 0, 'epochs')
```



SGD

```
[80] import tensorflow as tf
     tf.random.set_seed(50)
     model1= tf.keras.Sequential([
                                  tf.keras.layers.Dense(50),
                                  tf.keras.layers.Dense(40),
                                  tf.keras.layers.Dense(30),
                                  tf.keras.layers.Dense(20),
                                  tf.keras.layers.Dense(10),
                                  tf.keras.layers.Dense(5),
                                  tf.keras.layers.Dense(1)
     ])
```

```
[81] model1.compile(loss= tf.keras.losses.mae,
                    optimizer= tf.keras.optimizers.SGD(),
                    metrics= ["mae"])
```

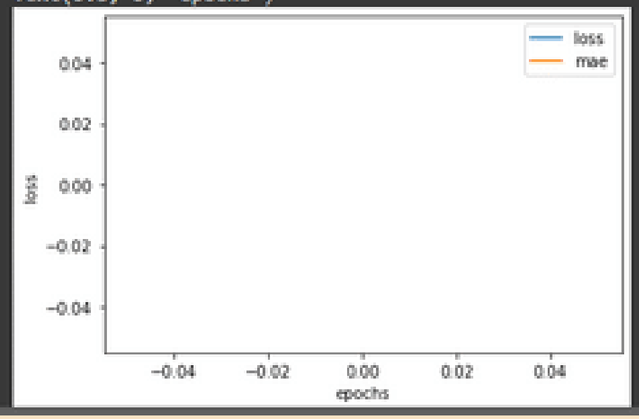
```
[82] history= model1.fit(X_train, y_train, epochs= 50, verbose=0)
```

```
[83] model1.evaluate(X_test, y_test)

     19/19 [==============================] - 0s 2ms/step - loss: nan - mae: nan
     [nan, nan]
```

```
[84] pd.DataFrame(history.history).plot()
     plt.ylabel("loss")
     plt.xlabel("epochs")

     Text(0.5, 0, 'epochs')
```



BAGGING

BAGGING REGRESSOR

```
[ ]  from sklearn import datasets
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import accuracy_score
     from sklearn.model_selection import cross_val_score
     from sklearn.ensemble import BaggingRegressor

[ ]  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=50)
```

```
[19]  clf = BaggingRegressor(n_estimators = 100, random_state = 0)

[20]  a = clf.fit(X_train, y_train)

      mae = metrics.mean_absolute_error(y_test, a.predict(X_test))
      mae

      84874.04109589041
```

DECISION TREE

```
[23] from sklearn.tree import DecisionTreeRegressor
```

```
[24] tree = DecisionTreeRegressor(max_depth=3, random_state=50)
```

```
[25] b = tree.fit(X_train, y_train)
```

```
mae = metrics.mean_absolute_error(y_test, b.predict(X_test))
mae
```

```
93247.65921985898
```

RANDOM FOREST

```
[28] from sklearn.ensemble import RandomForestRegressor

[29] rf = RandomForestRegressor(n_estimators = 100, random_state = 40)

[30] c = rf.fit(X_train, y_train)

     mae = metrics.mean_absolute_error(y_test, c.predict(X_test))
     mae

     85234.24657534246
```

BOOSTING

Ada Boost

```
[32] from sklearn.ensemble import AdaBoostRegressor

     adaboost = AdaBoostRegressor(n_estimators=500, learning_rate=0.01, random_state=0)

[33] d = adaboost.fit(X_train, y_train)

[34] mae = metrics.mean_absolute_error(y_test, d.predict(X_test))
     mae

     88188.91582677524
```

GBM

```
[65] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=50)
```

```
[66] from sklearn.ensemble import GradientBoostingRegressor

     grad_boost= GradientBoostingRegressor(max_depth=5,learning_rate=0.01,random_state=0,n_estimators=1000)
```

```
[67] e = grad_boost.fit(X_train, y_train)
```

```
[68] mae = metrics.mean_absolute_error(y_test, e.predict(X_test))
     mae
```

```
83447.8681144848
```

XGBoost

```
[39] import xgboost as xgb
     from xgboost import XGBRegressor
     xgb_boost=xgb.XGBRegressor(random_state=50,learning_rate=0.01,n_estimators=700)
```

```
[40] f = xgb_boost.fit(X_train, y_train)
```

```
[17:30:34] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now depr
```

```
[41] mae = metrics.mean_absolute_error(y_test, f.predict(X_test))
     mae
```

```
83556.74272260274
```

BAGGING

90 - 10 train-test split is giving the least MAE value.

90-10	84874.041
80-20	86866.267
75-25	87879.0
70-30	88789.349
60-40	88854.507