ENGINEERING GRADUATE SALARY

Group 12 | Bhavan's Vivekananda College
Y . Shan Koushik | Y . Sruthi | A.Rahul | V . Akash

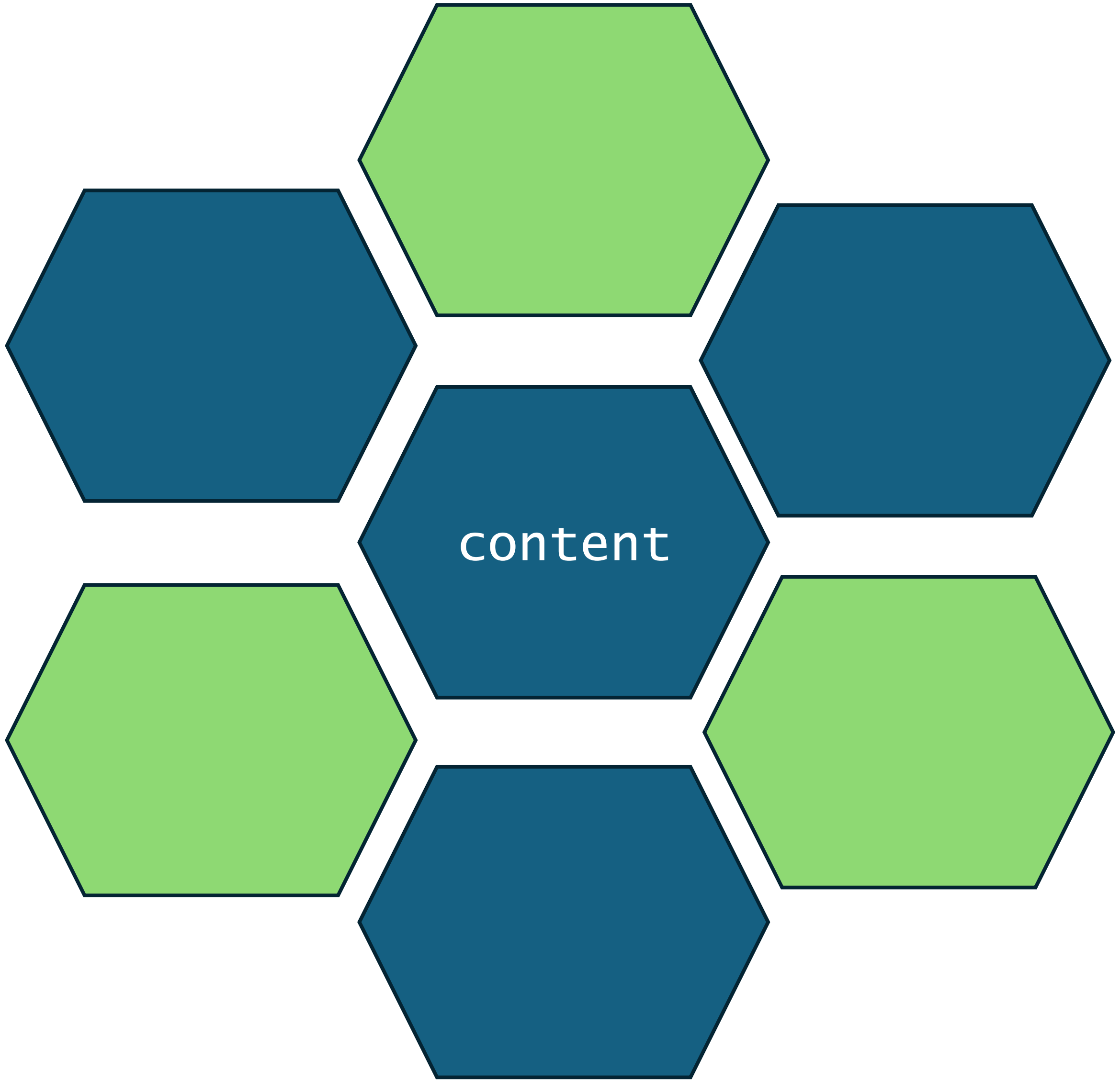
ABSTRACT

- This study aims to classify engineering graduates based on potential salary brackets using key demographic, academic, and experience-related features.
- Engineering graduates' salaries vary widely depending on factors like field, GPA, internship experience, and technical skills. Accurate salary classification can assist graduates in aligning their job expectations and aid recruiters in candidate selection.

OBJECTIVE

- Develop a machine learning model to classify graduates into salary brackets (e.g., low, medium, high) based on specific input features.
- A reliable model that classifies graduates into salary categories with high accuracy, providing valuable insights for educational institutions, students, and employers.

CONTENT



INTRODUCTION

- **Context and Background:** Engineering graduates' salaries often depend on diverse factors, including GPA, skill set, internship experience, and chosen specialization.

- **Dataset Details:**

Utilizes a dataset with features such as GPA, work experience, field of engineering, and specific technical skills. The target variable is the salary category, divided into classes (low, medium, high).

LITERATURE REVIEW

LITERATURE REVIEW

Engineering Graduate Salary Prediction: A Data-Driven Approach Using Linear Regression

1. Ajay Talele,
2. Shripad Wattamwar,
3. Ranjeet Thopte,
4. Onkar Waghmode,
5. Vasu Mahajan

- **Accuracy of Predictions:** The Engineering Graduate Salary Prediction System employs linear regression to provide reasonably accurate salary forecasts, as evidenced by the close alignment of predicted and actual salaries. Limitations include outliers and reliance on historical data (2007-2017), affecting real-time relevance.

- **Future Improvements:** Enhancing model precision could involve incorporating factors like industry trends and economic indicators or adopting advanced techniques such as deep learning and natural language processing.

https://www.kuey.net/index.php/kuey/article/view/5125/3539

DATA PREPROCESSING

DATA

Dataset: Our Dataset Consists Of 32 Variables And 3000 Records

Source : https://drive.google.com/file/d/1CZBa0RBm88cfdQzSyLVkn6n9Peuo3fsm/view?usp=drivesdk

	ID	Gender	DOB	10percentage	10board	12graduation	12percentage	12board	CollegeID	CollegeTier	...	MechanicalEngg	ElectricalEngg	TelecomEngg	CivilEngg	conscientiousness	agreeable
0	604399	f	1990-10-22	87.80	cbse	2009	84.00	cbse	6920	1	...	-1	-1	-1	-1	-0.1590	0.
1	988334	m	1990-05-15	57.00	cbse	2010	64.50	cbse	6624	2	...	-1	-1	-1	-1	1.1336	0.
2	301647	m	1989-08-21	77.33	maharashtra state board,pune	2007	85.17	amravati divisional board	9084	2	...	-1	-1	260	-1	0.5100	-0.
3	582313	m	1991-05-04	84.30	cbse	2009	86.00	cbse	8195	1	...	-1	-1	-1	-1	-0.4463	0.
4	339001	f	1990-10-30	82.00	cbse	2008	75.00	cbse	4889	2	...	-1	-1	-1	-1	-1.4992	-0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2993	103174	f	1989-04-17	75.00	0	2005	73.00	0	1263	2	...	-1	-1	-1	-1	-1.1901	0.
2994	352811	f	1991-07-22	84.00	state board	2008	77.00	state board	9481	2	...	-1	-1	-1	-1	-0.1082	0.
2995	287070	m	1988-11-24	91.40	bsemp	2006	65.56	bsemp	547	2	...	-1	-1	-1	-1	-0.8810	0.
2996	317336	m	1988-08-25	88.64	karnataka education board	2006	65.16	karnataka education board	1629	2	...	-1	-1	-1	-1	1.4374	1.
2997	993701	m	1992-05-27	77.00	state board	2009	75.50	state board	1111	2	...	-1	-1	-1	-1	-0.5899	-1.

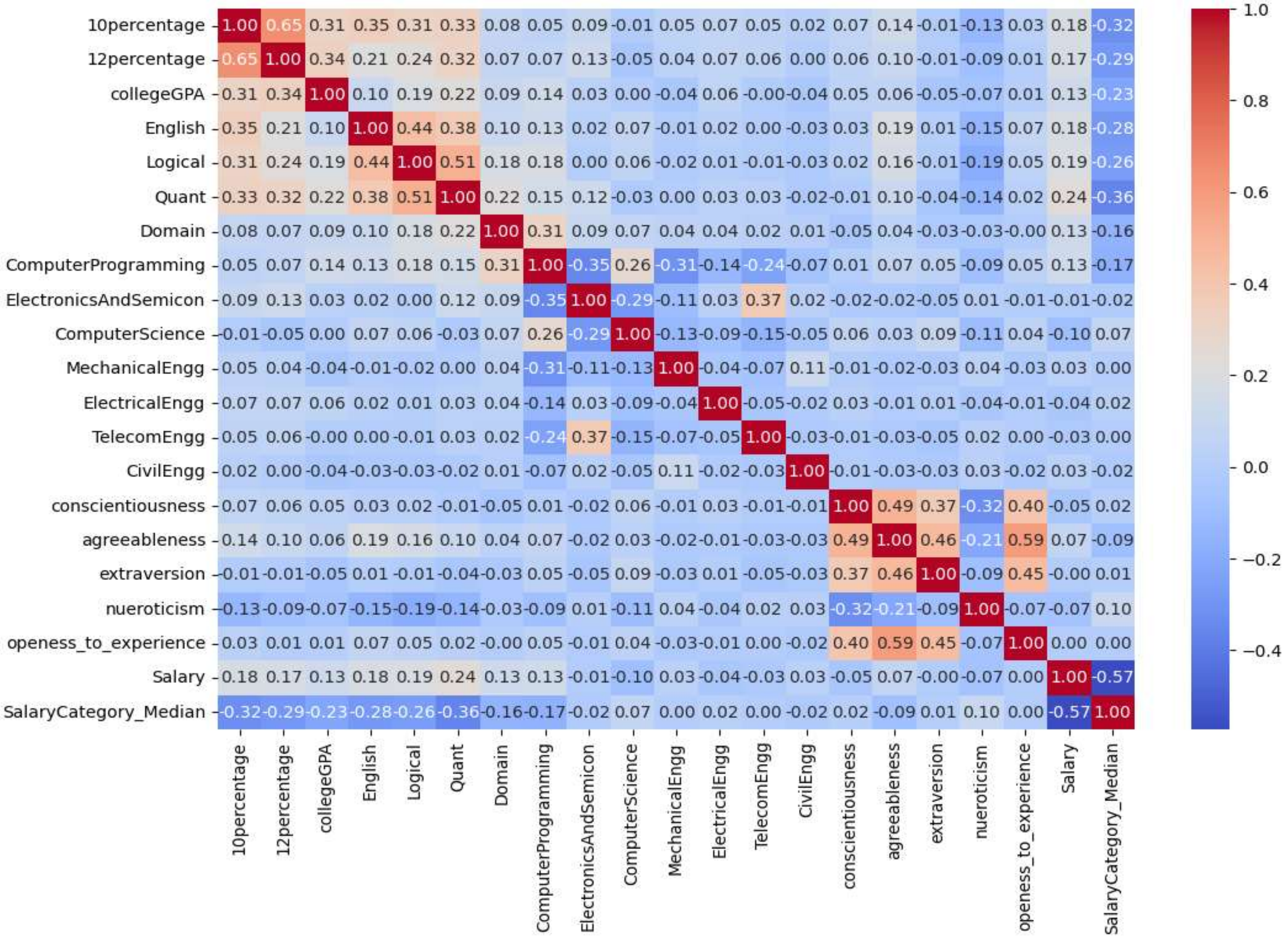
DATA CLEANING

- Checked for null /NaN values : We replaced the Nan values with mean and mode of the column based on the attribute type.
- Garbage values : We replaced the garbage values of the column based on the domain knowledge of the dataset.
- Dummification: enabled and dummified the categorical variables

CATEGORICAL VARIABLES	CONTINUOUS VARIABLES
GENDER	10 PERCENTAGE
SPECIALIZATION	12 PERCENTAGE
SALARY	COLLEGE CGPA

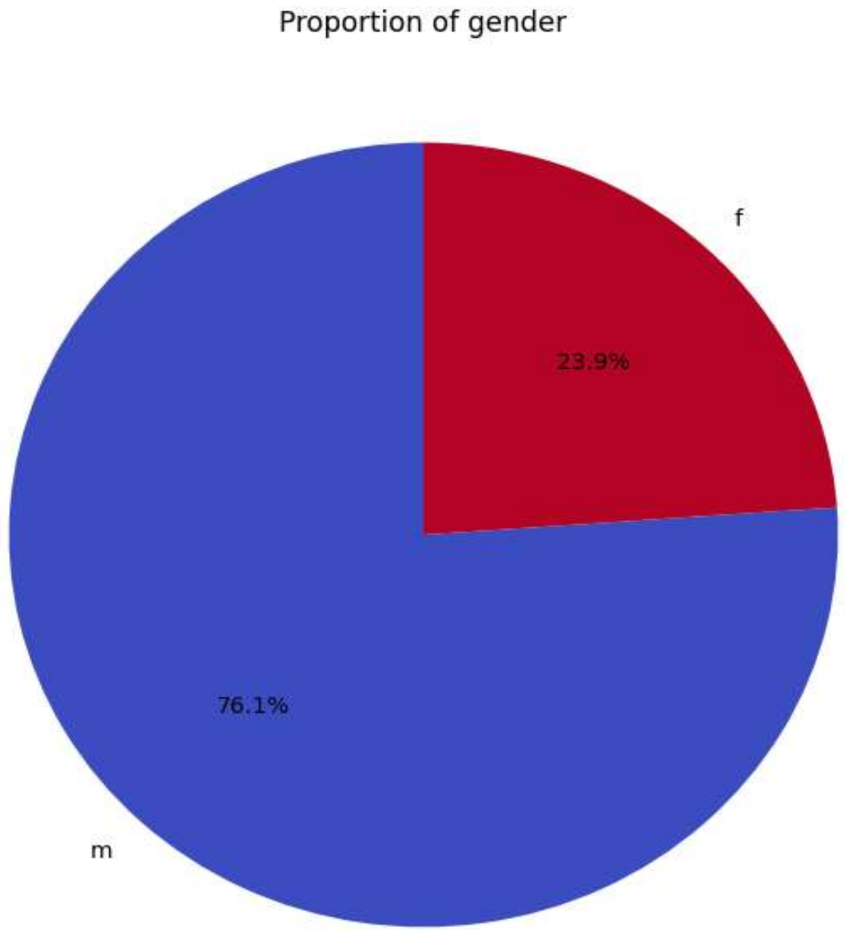
EXPLORATORY DATA ANALYSIS

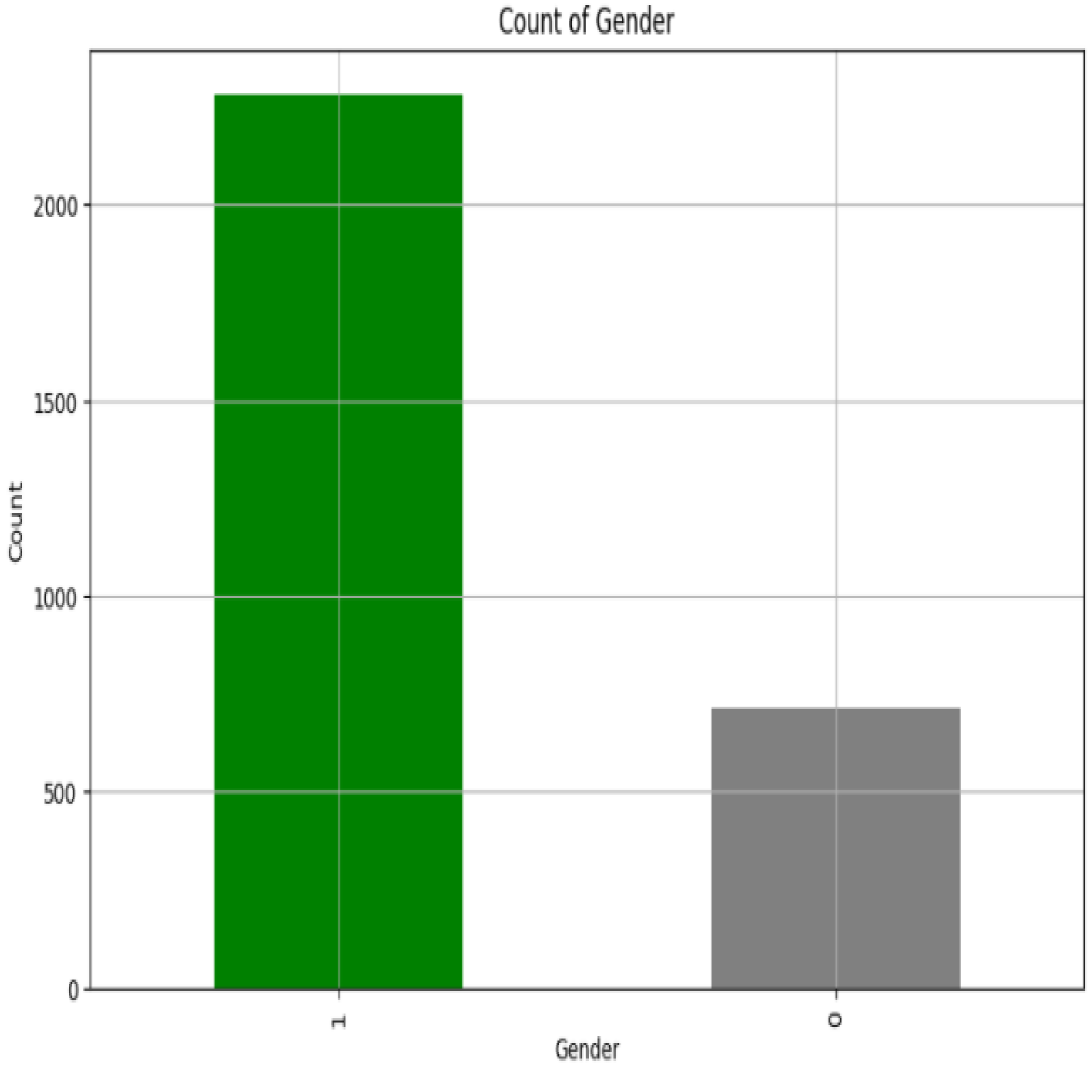
CORRELATION MATRIX



- The Most Positively Correlated Variables Are Domain-college Cgpa,10percentage-electronics & Computer Science-extraversion
- The Next Most Positively Correlated Variables Are Domain-10percentage

GENDER COUNT & PROPORTION



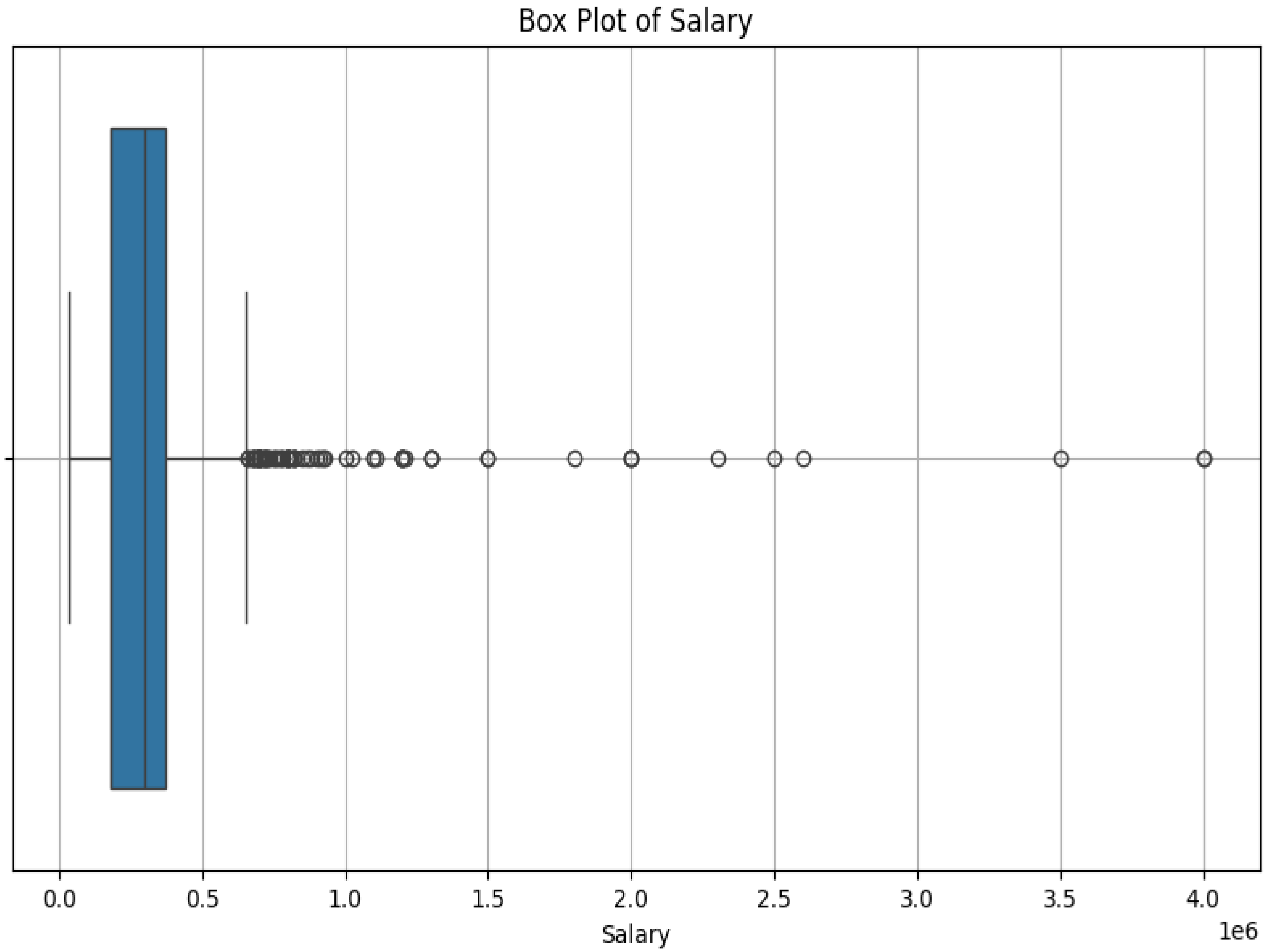


The Male Count Is More Than The Female Count

HISTOGRAM

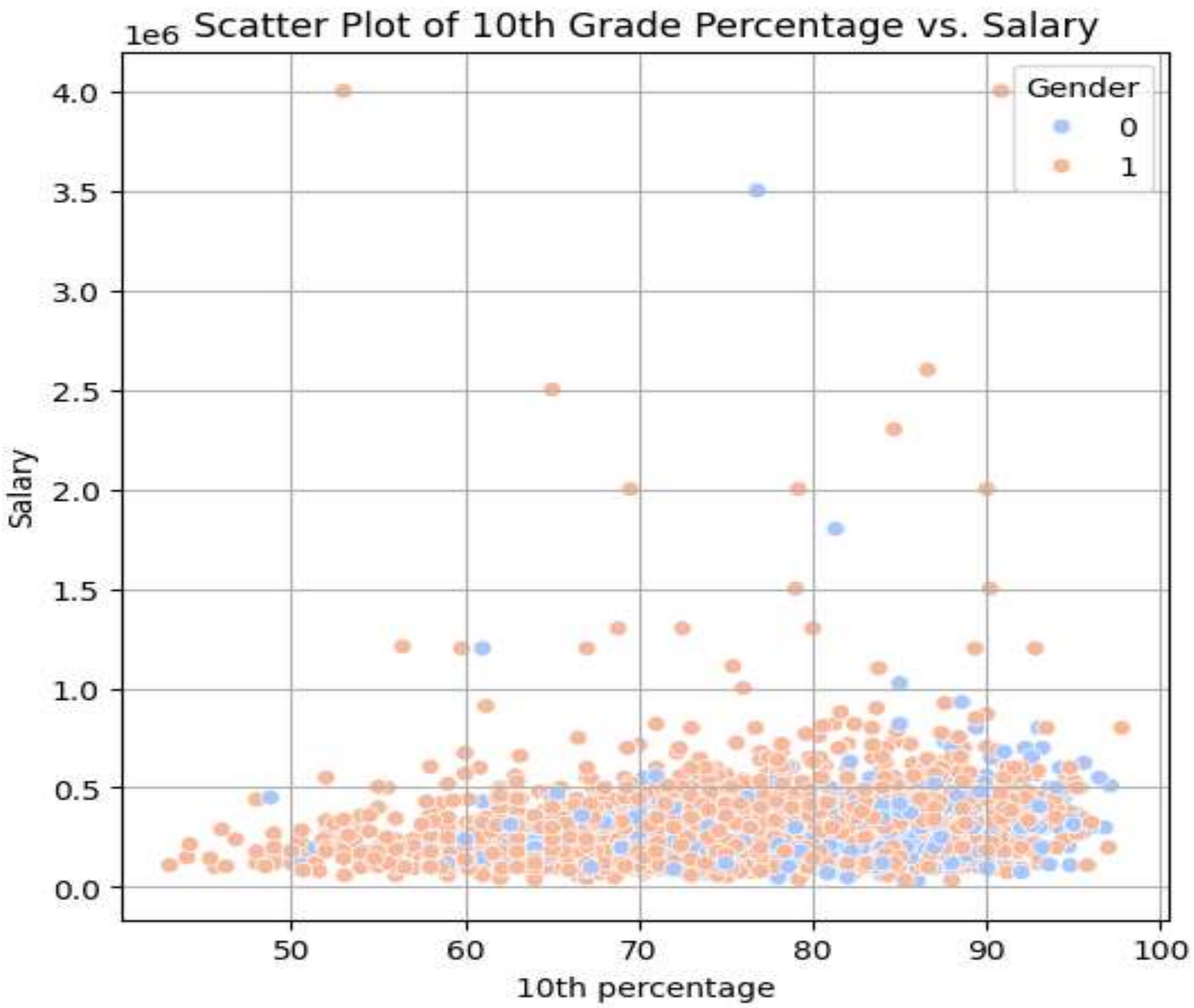
- The majority of salaries are concentrated at the lower end (below 500,000), as shown by the tall bars on the left.
- There are very few high salaries (over 1,000,000), as indicated by the much shorter bars or absence of bars on the right side.
- This distribution appears to be **right-skewed** or **positively skewed**, meaning most salaries are on the lower end with a few high outliers.

BOX PLOT



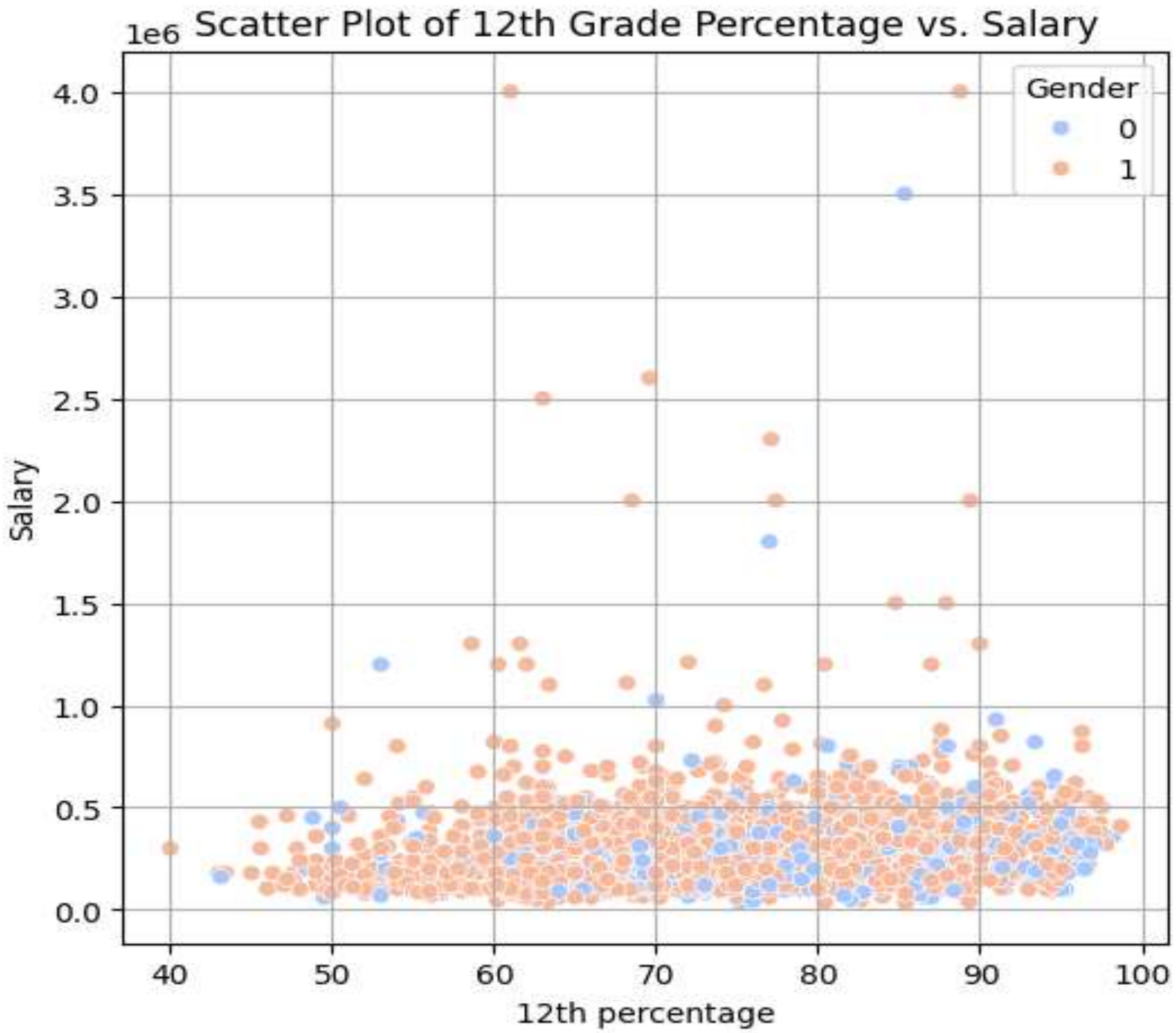
- Most salaries are tightly clustered in the lower range (left side), with a median well below 500,000
- There are numerous outliers on the right side (high salaries), extending up to about 4,000,000
- This distribution is highly **right-skewed**, as shown by the concentration of data at lower values and the long tail of outliers extending to higher values

SCATTER PLOT



- There appears to be a slight positive trend, where higher percentages in 10th grade tend to correlate with slightly higher salaries, though this trend is not strong.
- Both genders are fairly spread across the range of 10th grade percentages
- Many data points are clustered in the lower salary range (below 500,000), regardless of the 10th grade percentage

SCATTER PLOT



- There appears to be a slight positive trend, where higher percentages in 12th grade tend to correlate with slightly higher salaries, though this trend is not strong.
- Both genders are fairly spread across the range of 12th grade percentages
- Many data points are clustered in the lower salary range (below 500,000), regardless of the 12th grade percentage

MULTICOLLINEARITY CHECK

	variables	VIF
0	10percentage	2.0
1	12percentage	1.9
2	collegeGPA	1.3
3	English	1.4
4	Logical	1.6
...	...	...
63	Specialization_other	4.5
64	Specialization_telecommunication engineering	2.4
65	Specialization_Category_Electronics & Communic...	inf
66	Specialization_Category_Mechanical & Production	inf
67	Specialization_Category_Other	402.7

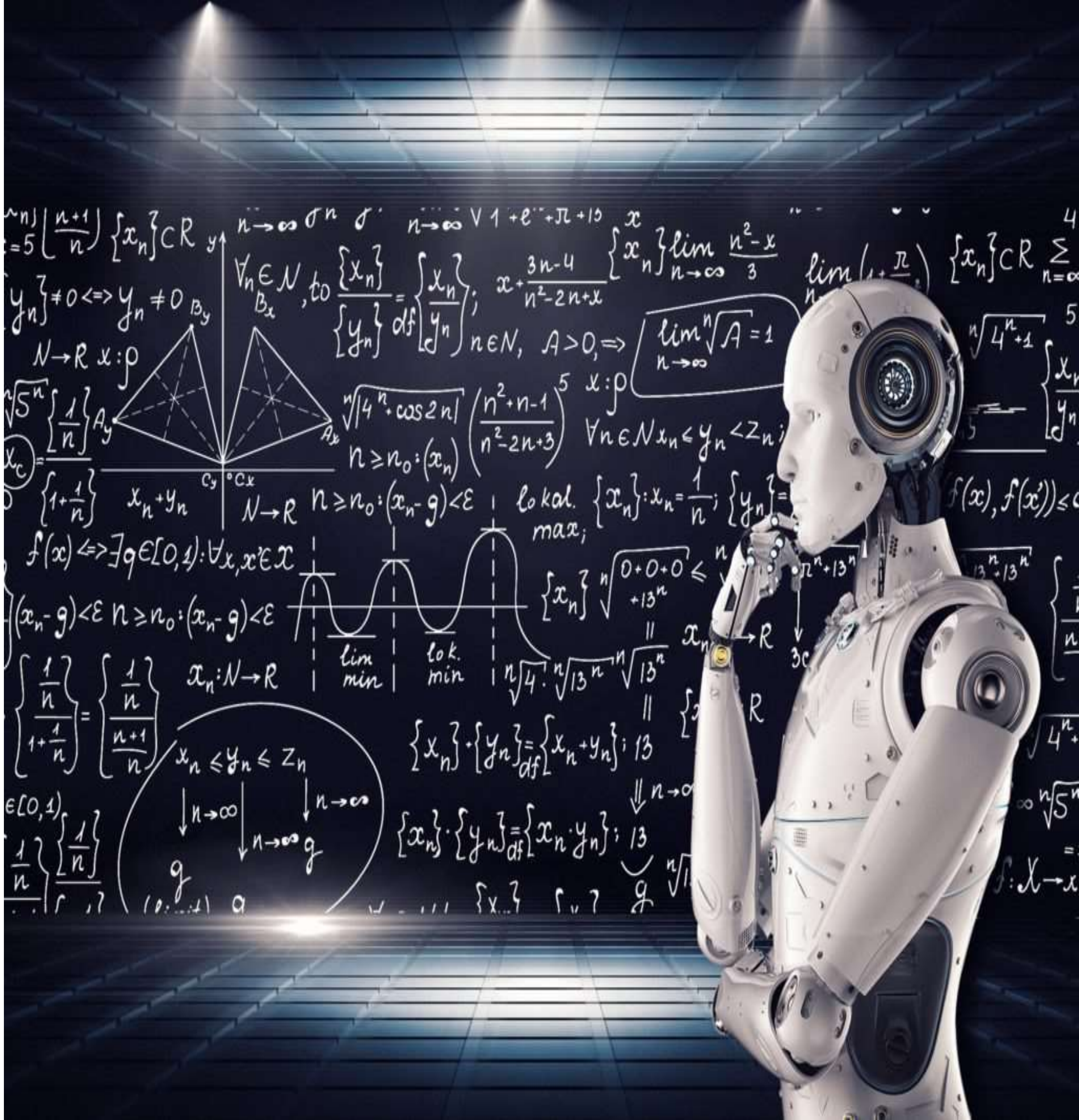
68 rows × 2 columns

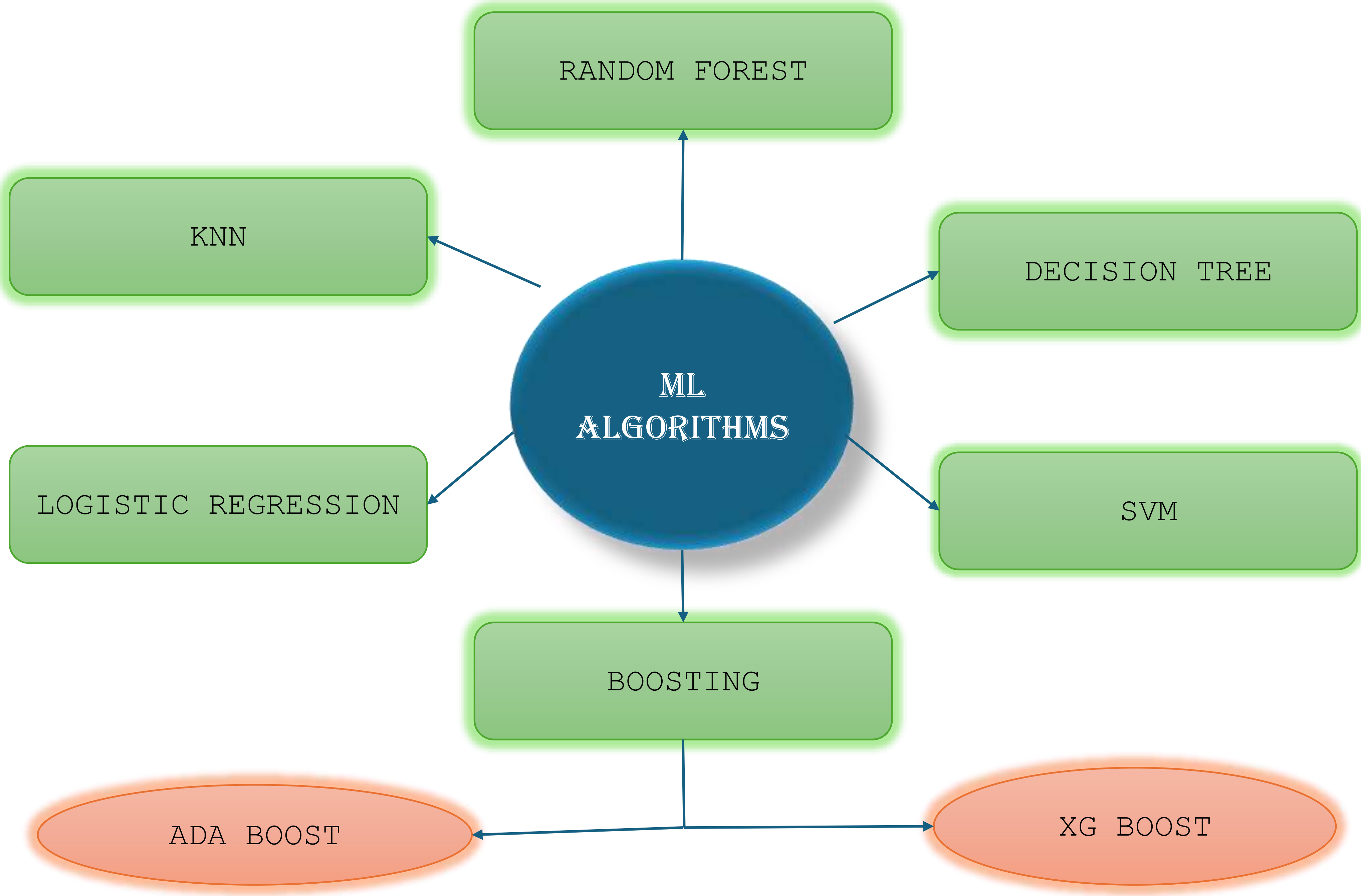
	variables	VIF
0	Quant	23.0
1	Domain	4.4
2	ComputerProgramming	7.9
3	ElectronicsAndSemicon	5.8
4	ComputerScience	1.9
5	MechanicalEngg	6.9
6	ElectricalEngg	2.0
7	TelecomEngg	1.5
8	CivilEngg	3.1
9	conscientiousness	1.6
10	agreeableness	2.0
11	extraversion	1.4
12	nueroticism	1.2
13	openess_to_experience	1.7
14	Salary	3.5
15	Gender_1	4.4
16	Degree_M.Sc. (Tech.)	1.0
17	Degree_M.Tech./M.E.	1.2
18	Degree_MCA	2.4
19	Specialization_applied electronics and instrum...	1.1
20	Specialization_automobile/automotive engineering	1.2
21	Specialization_biomedical engineering	1.0

	variables	VIF
0	Domain	2.7
1	ComputerScience	1.9
2	ElectricalEngg	1.8
3	TelecomEngg	1.5
4	CivilEngg	2.6
5	conscientiousness	1.5
6	agreeableness	2.0
7	extraversion	1.4
8	nueroticism	1.2
9	openess_to_experience	1.7
10	Degree_M.Sc. (Tech.)	1.0
11	Degree_M.Tech./M.E.	1.2
12	Degree_MCA	1.2
13	Specialization_applied electronics and instrum...	1.0
14	Specialization_automobile/automotive engineering	1.0
15	Specialization_biomedical engineering	1.0
16	Specialization_biotechnology	1.0
17	Specialization_ceramic engineering	1.0
18	Specialization_chemical engineering	1.0
19	Specialization_civil engineering	2.6
20	Specialization_computer and communication engi...	1.0
21	Specialization_computer engineering	1.4

Firstly The Variables With Infinity VIF Were Removed ,Then The Variables With (VIF>3) Were Removed

MACHINE LEARNING ALGORITHMS




RANDOM FOREST
KNN
DECISION TREE
ML
ALGORITHMS
LOGISTIC REGRESSION
SVM
BOOSTING
ADA BOOST
XG BOOST

60:40 TRAIN TEST SPLIT

ALGORITHMS	MODEL 1	MODEL 2
LOGISTIC REGRESSION	0.916	0.625
KNN	0.999	0.570
SVM	1	0.617
DECISION TREE	1	0.651
RANDOM FOREST	1	0.589
XG BOOST	1	0.610
ADA BOOST	1	0.597

70:30 TRAIN TEST SPLIT

ALGORITHMS	MODEL 1	MODEL 2
LOGISTIC REGRESSION	0.914	0.637
KNN	0.998	0.561
SVM	1	0.610
DECISION TREE	1	0.674
RANDOM FOREST	1	0.573
XG BOOST	1	0.610
ADA BOOST	1	0.601

75:25 TRAIN TEST SPLIT

ALGORITHMS	MODEL 1	MODEL 2
LOGISTIC REGRESSION	0.920	0.629
KNN	0.998	0.577
SVM	1	0.596
DECISION TREE	1	0.668
RANDOM FOREST	1	0.579
XG BOOST	1	0.590
ADA BOOS	1	0.581

80:20 TRAIN TEST SPLIT

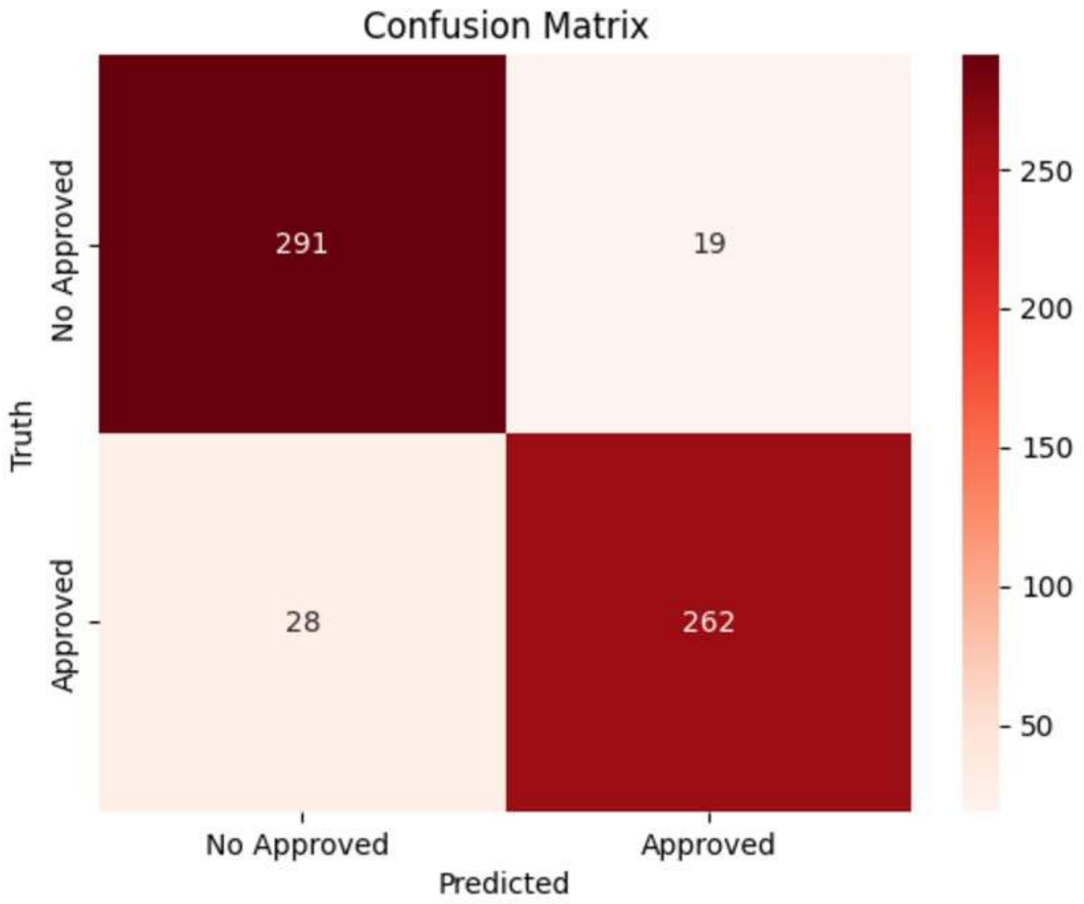
ALGORITHMS	MODEL 1	MODEL 2
LOGISTIC REGRESSION	0.922	0.618
KNN	0.998	0.598
SVM	1	0.597
DECISION TREE	1	0.667
RANDOM FOREST	1	0.602
XG BOOST	1	0.596
ADA BOOST	1	0.598

ALGORITHMS COMPARISION

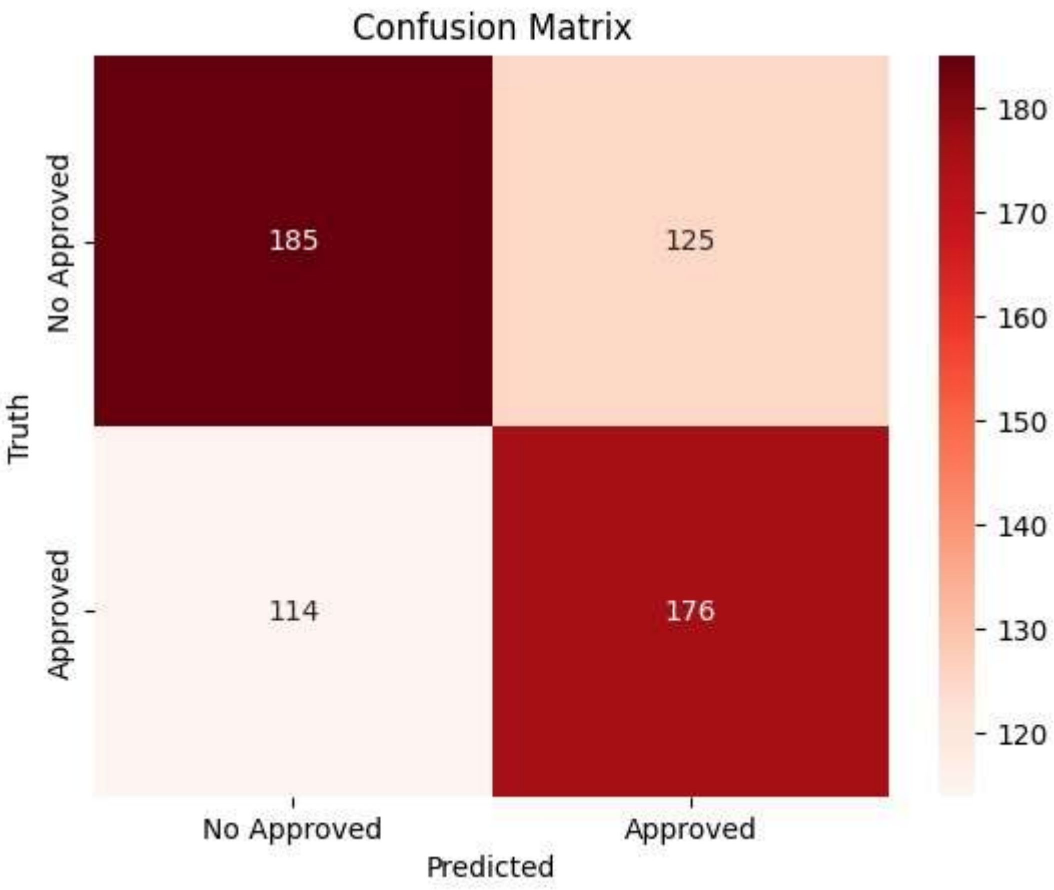
ALGORITHMS	MODEL 1	MODEL 2
LOGISTIC REGRESSION	0.922	0.618
KNN	0.998	0.598
SVM	1	0.597
DECISION TREE	1	0.667
RANDOM FOREST	1	0.602
XG BOOST	1	0.596
ADA BOOST	1	0.598

CONFUSION MATRIX

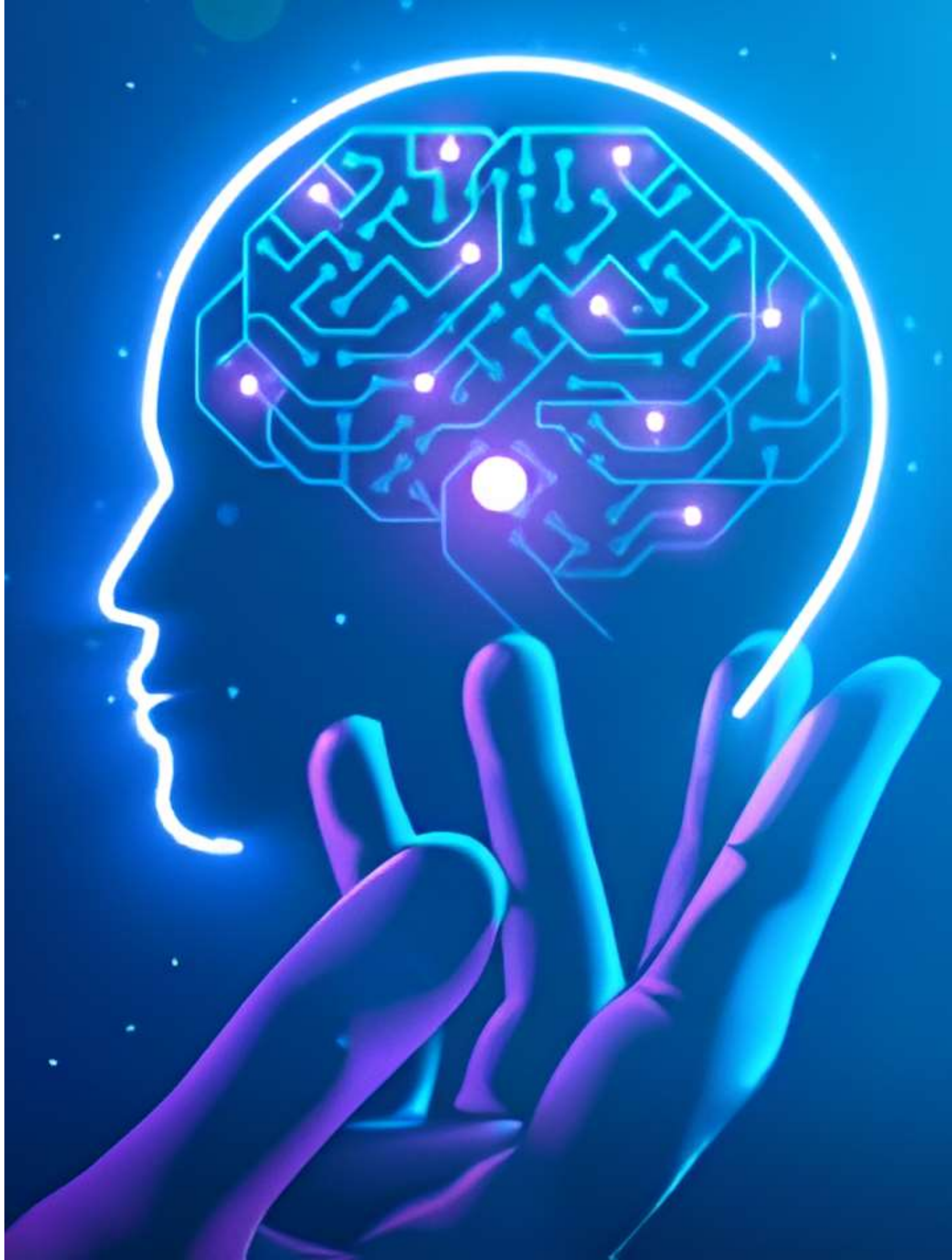
1.Logistic regression before VIF for 80:20 split



2.Decision tree after VIF for 80:20 spi

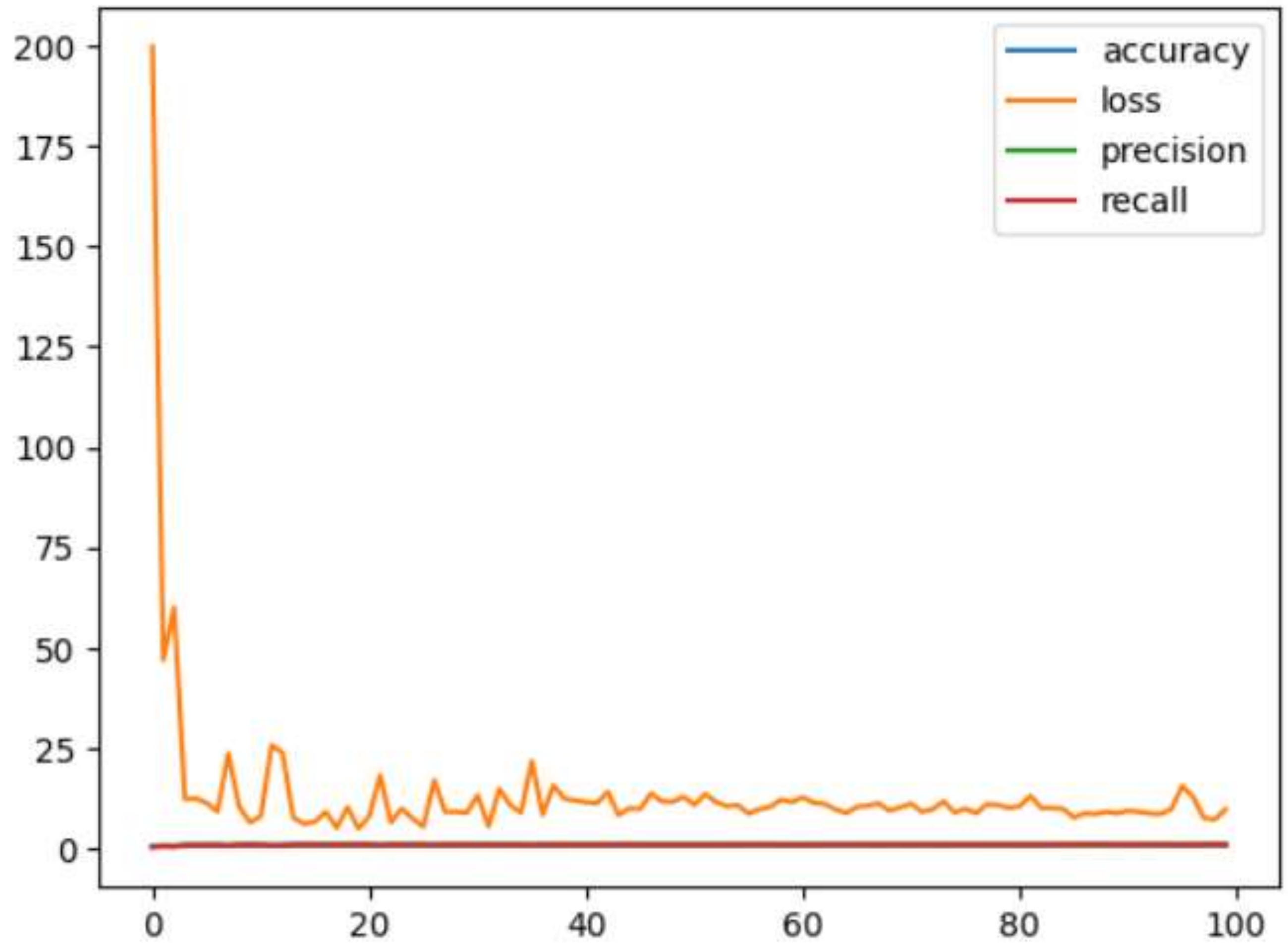


NEURAL NETWORK



Train test	Architecture	Optimizer	Epochs	Accuracy
60-40	30-20-10-1	Adam	100	0.7739
60-40	30-20-10-1	Adam	100	0.5804
70-30	30-20-10-1	Adam	100	0.5112
70-30	30-20-10-1	Adam	100	0.5597
75-25	30-20-10-1	Adam	100	0.6935
75-25	30-20-10-1	Adam	100	0.5547
80-20	30-20-10-1	Adam	100	0.9085
80-20	30-20-10-1	Adam	100	0.5620

NEURAL NETWORK PLOT



Train Test Split	80-20
Architecture	30-20-10-1
Optimizer	Adam
Epochs	100

SUMMARY

- The main aim of this research is to predict the salary of the engineering graduate's based on the performance of their education.
- After prediction we conclude that logistic regression is the best algorithm for model 1 i.e BEFORE VIF.
- The best split is 80:20

WORK DISTRIBUTION

TEAM MEMBER	WORK DONE
A RAHUL	COLLECTED REQUIRED INFORMATION AND DATA
AKASH	DATA PRE PROCESSING
Y SRUTHI	EXPLORATORY DATA ANALYSIS
SHAN KOUSHIK	IMPLMENTATION OF ML ALGORITHMS

Thank You

Colab Notebook Link

SHAN KOUSHIK
A RAHUL
Y SRUTHI
V AKASH

APPENDIX.

LOADING THE DATASET

```
data= pd.read_csv('Engineering_graduate_salary.csv')
data
```

	ID	Gender	DOB	10percentage	10board	12graduation	12percentage	12board	CollegeID	CollegeTier	...	MechanicalEngg	ElectricalEngg	TelecomEngg	CivilEngg	conscientiousness	agreeable
0	604399	f	1990-10-22	87.80	cbse	2009	84.00	cbse	6920	1	...	-1	-1	-1	-1	-0.1590	0.3
1	988334	m	1990-05-15	57.00	cbse	2010	64.50	cbse	6624	2	...	-1	-1	-1	-1	1.1336	0.0
2	301647	m	1989-08-21	77.33	maharashtra state board,pune	2007	85.17	amravati divisional board	9084	2	...	-1	-1	260	-1	0.5100	-0.1
3	582313	m	1991-05-04	84.30	cbse	2009	86.00	cbse	8195	1	...	-1	-1	-1	-1	-0.4463	0.2
4	339001	f	1990-10-30	82.00	cbse	2008	75.00	cbse	4889	2	...	-1	-1	-1	-1	-1.4992	-0.7
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2993	103174	f	1989-04-17	75.00	0	2005	73.00	0	1263	2	...	-1	-1	-1	-1	-1.1901	0.9
2994	352811	f	1991-07-22	84.00	state board	2008	77.00	state board	9481	2	...	-1	-1	-1	-1	-0.1082	0.0
2995	287070	m	1988-11-24	91.40	bsemp	2006	65.56	bsemp	547	2	...	-1	-1	-1	-1	-0.8810	0.1
2996	317336	m	1988-08-25	88.64	karnataka education board	2006	65.16	karnataka education board	1629	2	...	-1	-1	-1	-1	1.4374	1.2
2997	993701	m	1992-05-27	77.00	state board	2009	75.50	state board	1111	2	...	-1	-1	-1	-1	-0.5899	-1.9

NULL VALUES

```
data.isna().sum()
```

	0
Gender	0
10percentage	0
12percentage	0
Degree	0
Specialization	0
collegeGPA	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

CHECKING FOR DATA TYPES

```
[ ] data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2998 entries, 0 to 2997
Data columns (total 23 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Gender                 2998 non-null   object
 1   10percentage           2998 non-null   float64
 2   12percentage           2998 non-null   float64
 3   Degree                 2998 non-null   object
 4   Specialization         2998 non-null   object
 5   collegeGPA             2998 non-null   float64
 6   English                2998 non-null   int64
 7   Logical                2998 non-null   int64
 8   Quant                  2998 non-null   int64
 9   Domain                 2998 non-null   float64
 10  ComputerProgramming    2998 non-null   int64
 11  ElectronicsAndSemicon  2998 non-null   int64
 12  ComputerScience        2998 non-null   int64
 13  MechanicalEngg         2998 non-null   int64
 14  ElectricalEngg         2998 non-null   int64
 15  TelecomEngg            2998 non-null   int64
 16  CivilEngg              2998 non-null   int64
 17  conscientiousness      2998 non-null   float64
 18  agreeableness          2998 non-null   float64
 19  extraversion           2998 non-null   float64
 20  nueroticism            2998 non-null   float64
 21  openess_to_experience  2998 non-null   float64
 22  Salary                 2998 non-null   int64
dtypes: float64(9), int64(11), object(3)
memory usage: 538.8+ KB
```

Enlabling the Variables

```
[ ] df= data.replace(to_replace='m', value='1')
    df= df.replace(to_replace='f', value='0')
    df
```

	Gender	10percentage	12percentage	Degree	Specialization	collegeGPA	English	Logical	Quant	Domain
0	0	87.80	84.00	B.Tech/B.E.	instrumentation and control engineering	73.82	650	665	810	0.694479
1	1	57.00	64.50	B.Tech/B.E.	computer science & engineering	65.00	440	435	210	0.342315
2	1	77.33	85.17	B.Tech/B.E.	electronics & telecommunications	61.94	485	475	505	0.824666
3	1	84.30	86.00	B.Tech/B.E.	computer science & engineering	80.40	675	620	635	0.990009
4	0	82.00	75.00	B.Tech/B.E.	biotechnology	64.30	575	495	365	0.278457
...	...	...	...	...	...	...	...	...	...	...
2993	0	75.00	73.00	B.Tech/B.E.	electronics and communication engineering	70.00	505	485	445	0.538387
2994	0	84.00	77.00	B.Tech/B.E.	information technology	75.20	345	585	395	0.190153
2995	1	91.40	65.56	B.Tech/B.E.	information technology	73.19	385	425	485	0.600057
2996	1	88.64	65.16	B.Tech/B.E.	computer engineering	74.81	465	645	505	0.901490
2997	1	77.00	75.50	B.Tech/B.E.	information technology	69.30	370	390	285	0.486747

2998 rows × 25 columns

DIVIDING THE DATA SET

```
for i in range(data.shape[1]):
  print(data.iloc[:,i].unique())
  print(data.iloc[:,i].value_counts())
```

```
['f' 'm']
Gender
m     2282
f      716
Name: count, dtype: int64
[87.8  57.   77.33 84.3  82.   83.16 72.5  77.   76.8  81.2  85.   90.
 86.4  84.13 81.7  86.   66.15 79.29 60.   58.4  61.   50.   67.06 67.
 73.   86.17 78.   71.8  66.66 83.6  61.69 80.13 82.5  63.5  64.   76.
 91.   65.   70.16 74.6  66.5  78.4  62.   52.93 70.2  93.   53.4  84.2
 71.5  81.5  63.   74.   91.6  87.5  78.5  79.5  71.   66.   89.44 90.33
 76.5  70.   89.5  56.4  88.67 58.33 85.92 88.8  73.8  81.4  88.1  82.3
 66.7  72.   58.56 58.   75.6  75.   92.2  84.5  89.4  76.2  85.33 45.6
 86.5  75.83 69.4  85.6  80.6  69.   89.56 83.2  51.   60.7  90.6  75.4
 81.8  75.85 89.2  93.8  76.66 90.4  90.8  82.67 94.16 61.73 87.7  88.
 80.8  77.6  87.   89.8  80.   84.   89.6  59.57 83.   67.8  82.4  79.
 73.5  89.   87.4  93.33 71.3  81.   55.   83.4  64.8  83.5  82.27 71.1
 91.4  87.33 73.94 79.8  92.   78.93 52.7  69.5  67.25 88.2  67.2  56.
 90.2  83.68 84.4  85.8  83.04 79.2  77.86 81.66 82.6  91.2  62.4  72.4
 78.33 68.   86.6  61.6  85.83 69.66 79.78 91.86 79.66 84.67 64.4  71.66
 82.2  76.6  85.3  68.6  79.4  72.3  75.38 87.2  57.67 80.33 55.6  89.33
 86.3  73.2  70.3  65.2  72.17 84.6  80.07 92.47 66.33 88.64 75.86 88.66
 74.2  94.   72.2  59.   92.48 89.17 69.53 58.5  81.16 53.8  52.   76.7
 83.8  71.33 93.2  90.06 89.42 77.57 92.5  78.15 63.6  81.33 69.8  90.1
 91.8  64.2  87.63 80.16 92.6  80.3  76.48 93.6  79.6  86.83 89.76 73.4
 78.6  88.6  53.06 85.72 78.88 84.8  91.21 86.7  78.3  54.83 55.3  61.2
 67.36 61.75 55.33 91.1  75.52 85.5  86.08 87.6  80.2  65.26 70.1  85.2
 49.   77.8  74.3  68.2  87.62 93.4  82.28 64.56 69.33 91.04 75.12 64.5
 78.8  66.6  74.5  71.6  74.4  86.15 73.37 70.25 77.4  86.1  72.6  90.5
 89.12 81.86 62.15 67.12 91.84 70.6  56.16 66.85 56.78 68.33 78.2  78.61
 94.4  67.6  58.6  83.89 78.67 67.3  73.6  91.5  68.3  85.4  90.83 86.09
 84.83 83.66 72.8  82.25 61.57 86.14 74.88 88.3  68.14 67.72 87.58 88.5
 87.52 62.5  81.03 88.4  88.76 54.4  63.33 70.33 92.8  90.01 50.6  81.6
```

DROPPING COLUMNS

```
[ ]  data.columns
```

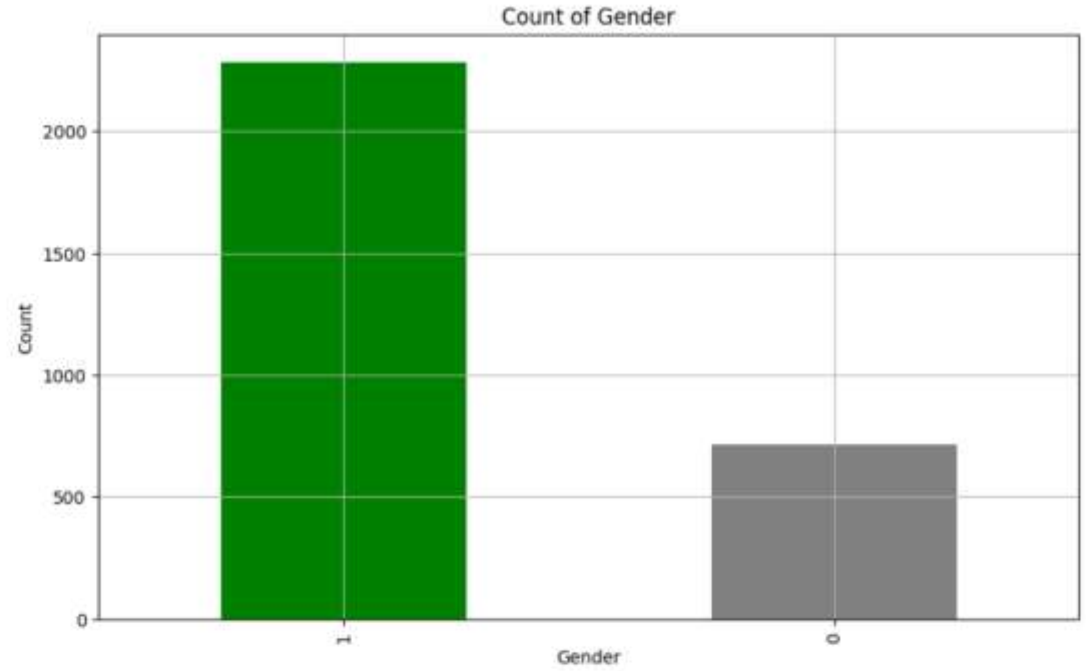
```
Index(['ID', 'Gender', 'DOB', '10percentage', '10board', '12graduation',
       '12percentage', '12board', 'CollegeID', 'CollegeTier', 'Degree',
       'Specialization', 'collegeGPA', 'CollegeCityID', 'CollegeCityTier',
       'CollegeState', 'GraduationYear', 'English', 'Logical', 'Quant',
       'Domain', 'ComputerProgramming', 'ElectronicsAndSemicon',
       'ComputerScience', 'MechanicalEngg', 'ElectricalEngg', 'TelecomEngg',
       'CivilEngg', 'conscientiousness', 'agreeableness', 'extraversion',
       'nueroticism', 'openess_to_experience', 'Salary'],
      dtype='object')
```

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

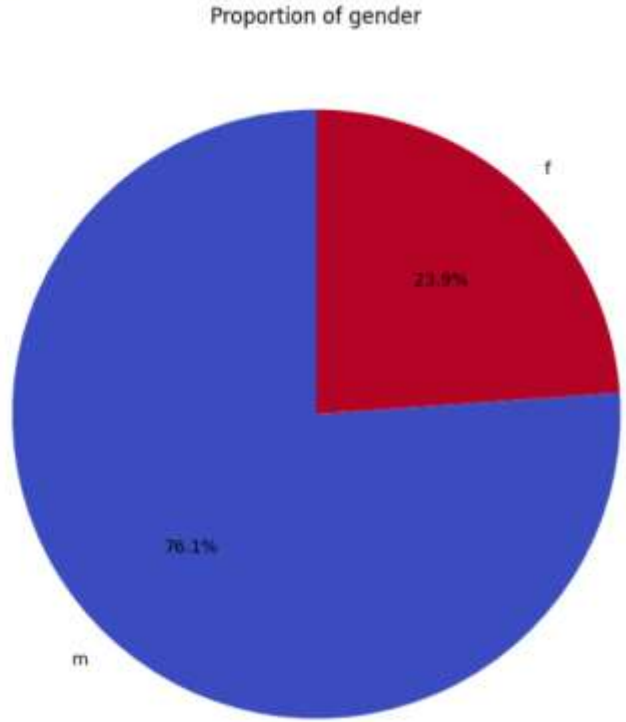
```
     Gender  10percentage  12percentage     Degree  \
0         f         87.80         84.00  B.Tech/B.E.
1         m         57.00         64.50  B.Tech/B.E.
2         m         77.33         85.17  B.Tech/B.E.
3         m         84.30         86.00  B.Tech/B.E.
4         f         82.00         75.00  B.Tech/B.E.
...     ...           ...           ...          ...
2993      f         75.00         73.00  B.Tech/B.E.
2994      f         84.00         77.00  B.Tech/B.E.
2995      m         91.40         65.56  B.Tech/B.E.
2996      m         88.64         65.16  B.Tech/B.E.
2997      m         77.00         75.50  B.Tech/B.E.

                                Specialization  collegeGPA  English  Logical  \
0       instrumentation and control engineering      73.82      650      665
1                computer science & engineering      65.00      440      435
2              electronics & telecommunications      61.94      485      475
3                computer science & engineering      80.40      675      620
4                                biotechnology      64.30      575      495
...                                        ...        ...      ...      ...
2993  electronics and communication engineering      70.00      505      485
2994                    information technology      75.20      345      585
2995                    information technology      73.19      385      425
2996                      computer engineering      74.81      465      645
2997                    information technology      69.30      370      390

      Quant    Domain  ...  MechanicalEngg  ElectricalEngg  TelecomEngg  \
0       810  0.694479  ...              -1              -1           -1
1       210  0.342315  ...              -1              -1           -1
2       505  0.824666  ...              -1              -1          260
3       635  0.990009  ...              -1              -1           -1
4       365  0.278457  ...              -1              -1           -1
...     ...       ...  ...             ...             ...          ...
2993    445  0.538387  ...              -1              -1           -1
2994    395  0.190153  ...              -1              -1           -1
2995    485  0.600057  ...              -1              -1           -1
2996    505  0.901490  ...              -1              -1           -1
2997    285  0.486747  ...              -1              -1           -1

      CivilEngg  conscientiousness  agreeableness  extraversion  nueroticism  \
0            -1            -0.1590         0.3789        1.2396      0.14590
```

BAR & PIE PLOTS

```python
# Bar Plot for Gender
plt.figure(figsize=(10, 6))
df['Gender'].value_counts().plot(kind='bar', color=['green', 'grey'])
plt.title('Count of Gender')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.grid(True)
plt.show()
```




```python
plt.figure(figsize=(8, 8))
data['Gender'].value_counts().plot.pie(autopct='%1.1f%%', startangle=90, cmap='coolwarm')
plt.title('Proportion of gender')
plt.ylabel('')
plt.show()
```



LOGISTIC REGRESSION BEFORE VIF

```
from sklearn.model_selection import train_test_split
X_train1, X_test1, y_train1, y_test1 = train_test_split(X, y, test_size=0.40, random_state=42)
X_train2, X_test2, y_train2, y_test2 = train_test_split(X, y, test_size=0.30, random_state=42)
X_train3, X_test3, y_train3, y_test3 = train_test_split(X, y, test_size=0.25, random_state=42)
X_train4 ,X_test4, y_train4, y_test4 = train_test_split(X, y, test_size=0.20, random_state=42)
```

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(C=1e9)
```

60-40

```
logreg.fit(X_train1, y_train1)
predictions = logreg.predict(X_test1)
print(predictions)
```

```
[1 1 1 ... 1 0 0]
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed t
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
z=confusion_matrix(y_test1, predictions)
print(z)
accuracy_score(y_test1,predictions)
```

```
[[605  30]
 [ 71 494]]
0.9158333333333334
```

KNN BEFORE VIF

60-40

```
[ ] from sklearn.neighbors import KNeighborsClassifier
```

```
model=KNeighborsClassifier(n_neighbors=25)
```

```
[ ] model.fit(X_train1, y_train1)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=25)
```

```
[ ] y_pred1 = model.predict(X_test1)
    y_pred1
```

```
array([1, 1, 1, ..., 1, 0, 0])
```

```
[ ] knn = pd.DataFrame({'Predicted':y_pred1,'Actual':y_test1})
    knn
```

	Predicted	Actual
1376	1	1
932	1	1
144	1	1
1752	0	0
51	0	0
...	...	...
308	1	1

SVM BEFORE VIF

60-40

```
[ ] from sklearn.svm import SVC
```

```
[ ] model1 = SVC(kernel='linear')
```

```
model1.fit(X_train1, y_train1)
```

SVC

SVC(kernel='linear')

```
[ ] y_pred1 = model1.predict(X_test1)
```

```
[ ] y_pred1
```

array([1, 1, 1, ..., 1, 0, 0])

```
[ ] svm = pd.DataFrame({'Predicted':y_pred1,'Actual':y_test1})
    svm
```

	Predicted	Actual
1376	1	1
932	1	1
144	1	1
1752	0	0
51	0	0
...	...	...
308	1	1

DECISION TREE'S BEFORE VIF

60-40

```
[ ] from sklearn.tree import DecisionTreeClassifier
    from sklearn import metrics
```

```
[ ]
    clf = DecisionTreeClassifier()
    clf = clf.fit(X_train1,y_train1)
```

```
y_pred1 = clf.predict(X_test1)
```

```
[ ]
    print("Accuracy:",metrics.accuracy_score(y_test1, y_pred1))
```

```
Accuracy: 1.0
```

```
[ ] clf = DecisionTreeClassifier(criterion="entropy", max_depth=3)
    clf = clf.fit(X_train1,y_train1)
    y_pred1 = clf.predict(X_test1)
    print("Accuracy:",metrics.accuracy_score(y_test1, y_pred1))
```

```
Accuracy: 1.0
```

```
[ ] clf = DecisionTreeClassifier(criterion="gini", max_depth=2)
    clf = clf.fit(X_train1,y_train1)
    y_pred1 = clf.predict(X_test1)
    print("Accuracy:",metrics.accuracy_score(y_test1, y_pred1))
```

```
Accuracy: 1.0
```

```
[ ] clf = DecisionTreeClassifier(criterion="gini", max_depth=3)
    clf = clf.fit(X_train1,y_train1)
```

RANDOM FOREST BEFORE VIF

60-40

```
[ ] from sklearn.ensemble import RandomForestClassifier
    from sklearn.tree import plot_tree
```

```
rf = RandomForestClassifier()
```

```
[ ] rf.fit(X_train1,y_train1)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
[ ] y_pred1=rf.predict(X_test1)
    print("Accuracy:",accuracy_score(y_test1,y_pred1))
```

```
Accuracy: 1.0
```

```
[ ] print(classification_report(y_test1, y_pred1))
    print(confusion_matrix(y_test1, y_pred1))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       635
           1       1.00      1.00      1.00       565

    accuracy                           1.00      1200
   macro avg       1.00      1.00      1.00      1200
weighted avg       1.00      1.00      1.00      1200

[[635   0]
```

ADABOOST BEFORE VIF

Collapse 15 child cells under ADABOOST BEFORE VIF (Press <Shift> to also collapse sibling sections)

```
[ ] from sklearn.ensemble import AdaBoostClassifier
    from sklearn.ensemble import AdaBoostClassifier
    from sklearn.tree import DecisionTreeClassifier
```

```
[ ] # Replace 'base_estimator' with 'estimator'
    base_estimator = DecisionTreeClassifier(max_depth=3, random_state=0)
    adaboost = AdaBoostClassifier(estimator=base_estimator, # Changed argument name here
                                  n_estimators=3,random_state=0)
```

60-40

```
[ ] adaboost.fit(X_train1, y_train1)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_weight_boosting.py:527: FutureWarning: The S
  warnings.warn(
```

```
▸ AdaBoostClassifier
▸ estimator: DecisionTreeClassifier
  ▸ DecisionTreeClassifier
```

```
[ ] y_pred1 = adaboost.predict(X_test1)
    print("Accuracy:",accuracy_score(y_test1,y_pred1))
    print(classification_report(y_test1, y_pred1))
    print(confusion_matrix(y_test1, y_pred1))
```

```
Accuracy: 1.0
```

XGBOOST BEFORE VIF

```
import xgboost as xgb
```

```
model1 = xgb.XGBClassifier()
model2 = xgb.XGBClassifier(n_estimators=100, max_depth=8, learning_rate=0.1, subsample=0.5)
```

```
y_train1 = y_train1.astype('int')
y_train2 = y_train2.astype('int')
y_train3 = y_train3.astype('int')
y_train4 = y_train4.astype('int')
y_test1= y_test1.astype('int')
y_test2= y_test2.astype('int')
y_test3= y_test3.astype('int')
y_test4= y_test4.astype('int')
```

60-40

```
model1.fit(X_train1, y_train1)
model2.fit(X_train1,y_train1)
```

XGBClassifier

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.1, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=8, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=100, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)
```

```
pred1 = model1.predict(X_test1)
pred2 = model2.predict(X_test1)

print('Model 1 XGboost Report %r' % (classification_report(y_test1, pred1)))
print('Model 2 XGboost Report %r' % (classification_report(y_test1, pred2)))
```

VIF

```
from statsmodels.stats.outliers_influence import variance_inflation_factor

def calc_vif(X):

    # Calculating VIF
    vif = pd.DataFrame()
    vif["variables"] = X.columns
    vif["VIF"] = [variance_inflation_factor(X.values, i).round(1) for i in range(X.shape[1])]

    return(vif)

calc_vif(X)
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/stats/outliers_influence.py:197: RuntimeWarning: divide by zero encountered in scalar divide
  vif = 1. / (1. - r_squared_i)
```

	variables	VIF
0	10percentage	2.0
1	12percentage	1.9
2	collegeGPA	1.3
3	English	1.4
4	Logical	1.6
...	...	...
63	Specialization_other	4.5
64	Specialization_telecommunication engineering	2.4
65	Specialization_Category_Electronics & Communic...	inf
66	Specialization_Category_Mechanical & Production	inf
67	Specialization_Category_Other	402.7

68 rows × 2 columns

```
calc_vif(X.drop('Specialization_Category_Mechanical & Production', axis=1))
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/stats/outliers_influence.py:197: RuntimeWarning: divide by zero encountered in scalar divide
  vif = 1. / (1. - r_squared_i)
```

LOGISTIC REGRESSION AFTER VIF

60-40

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(C=1e9)

logreg.fit(X_train1_nomulti, y_train1_nomulti)
predictions1 = logreg.predict(X_test1_nomulti)
print(predictions1)
```

```
[0 0 1 ... 0 0 0]
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed to cc
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
z=confusion_matrix(y_test1_nomulti, predictions1)
z
```

```
array([[446, 189],
       [261, 304]])
```

```
accuracy_score(y_test1_nomulti,predictions1)
```

```
0.625
```

```
print(classification_report(y_test1_nomulti,predictions1))
```

```
              precision    recall  f1-score   support

           0       0.63      0.70      0.66       635
           1       0.62      0.54      0.57       565

    accuracy                           0.62      1200
```

KNN AFTER VIF

60-40

```
model.fit(X_train1_nomulti, y_train1_nomulti)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=25)
```

```
y_pred1_nomulti = model.predict(X_test1_nomulti)
y_pred1_nomulti
```

```
array([0, 0, 1, ..., 0, 0, 0])
```

```
knn = pd.DataFrame({'Predicted':y_pred1_nomulti,'Actual':y_test1_nomulti})
knn
```

	Predicted	Actual
1376	0	1
932	0	1
144	1	1
1752	1	0
51	0	0
...	...	...
308	0	1
2318	1	1
749	0	1
1431	0	0
1236	0	0

1200 rows × 2 columns

SVM AFTER VIF

60-40

```
model1 = SVC(kernel='linear')
```

```
model1.fit(X_train1_nomulti, y_train1_nomulti)
```

SVC

SVC(kernel='linear')

```
y_pred1_nomulti = model1.predict(X_test1_nomulti)
```

```
y_pred1_nomulti
```

array([0, 0, 1, ..., 0, 0, 0])

```
svm = pd.DataFrame({'Predicted':y_pred1_nomulti,'Actual':y_test1_nomulti})
svm
```

	Predicted	Actual
1376	0	1
932	0	1
144	1	1
1752	1	0
51	1	0
...	...	...
308	0	1
2318	1	1
749	0	1

DECISION TREE'S AFTER VIF

60-40

```
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
```

```
clf = DecisionTreeClassifier()
clf = clf.fit(X_train1_nomulti,y_train1_nomulti)
```

```
y_pred1_nomulti = clf.predict(X_test1_nomulti)
```

```
print("Accuracy:",metrics.accuracy_score(y_test1_nomulti, y_pred1_nomulti))
```

```
Accuracy: 0.5416666666666666
```

```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="entropy", max_depth=3)

# Train Decision Tree Classifer
clf = clf.fit(X_train1_nomulti,y_train1_nomulti)

#Predict the response for test dataset
y_pred1_nomulti = clf.predict(X_test1_nomulti)

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test1_nomulti, y_pred1_nomulti))
```

```
Accuracy: 0.5983333333333334
```

```
clf = DecisionTreeClassifier(criterion="gini", max_depth=2)

# Train Decision Tree Classifer
clf = clf.fit(X_train1_nomulti,y_train1_nomulti)
```

RANDOM FOREST AFTER VIF

60-40

```
rf.fit(X_train1_nomulti,y_train1_nomulti)
```

```
RandomForestClassifier
RandomForestClassifier()
```

```
[ ]
y_pred_train1_nomulti=rf.predict(X_test1_nomulti)
print("Accuracy:",accuracy_score(y_test1_nomulti,y_pred1_nomulti))
print(classification_report(y_test1_nomulti, y_pred1_nomulti))
print(confusion_matrix(y_test1_nomulti, y_pred1_nomulti))
```

```
Accuracy: 0.5908333333333333
              precision    recall  f1-score   support

           0       0.63      0.54      0.58       635
           1       0.56      0.64      0.60       565

    accuracy                           0.59      1200
   macro avg       0.59      0.59      0.59      1200
weighted avg       0.60      0.59      0.59      1200

[[346 289]
 [202 363]]
```

70-30

```
[ ]
rf.fit(X_train2_nomulti,y_train2_nomulti)
```

```
RandomForestClassifier
RandomForestClassifier()
```

```
[ ]
y_pred_train2_nomulti=rf.predict(X_test2_nomulti)
print("Accuracy:",accuracy_score(y_test2_nomulti,y_pred2_nomulti))
print(classification_report(y_test2_nomulti, y_pred2_nomulti))
print(confusion_matrix(y_test2_nomulti, y_pred2_nomulti))
```

ADABOOST AFTER VIF

60-40

```
adaboost.fit(X_train1_nomulti, y_train1_nomulti)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_weight_boosting.py:527: FutureWarning: The SAMME.R algorithm
  warnings.warn(
```

```
AdaBoostClassifier
estimator: DecisionTreeClassifier
DecisionTreeClassifier
```

```
y_pred1_nomulti = adaboost.predict(X_test1_nomulti)
print("Accuracy:",accuracy_score(y_test1_nomulti,y_pred1_nomulti))
print(classification_report(y_test1_nomulti, y_pred1_nomulti))
print(confusion_matrix(y_test1_nomulti, y_pred1_nomulti))
```

```
Accuracy: 0.5975
              precision    recall  f1-score   support

           0       0.62      0.60      0.61       635
           1       0.57      0.59      0.58       565

    accuracy                           0.60      1200
   macro avg       0.60      0.60      0.60      1200
weighted avg       0.60      0.60      0.60      1200

[[382 253]
 [230 335]]
```

70-30

```
adaboost.fit(X_train2_nomulti, y_train2_nomulti)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_weight_boosting.py:527: FutureWarning: The SAMME.R algorithm
  warnings.warn(
```

```
AdaBoostClassifier
```

XGBOOST AFTER VIF

```
[ ] y_train1_nomulti = y_train1_nomulti.astype('int')
    y_train2_nomulti = y_train2_nomulti.astype('int')
    y_train3_nomulti = y_train3_nomulti.astype('int')
    y_train4_nomulti = y_train4_nomulti.astype('int')
    y_test1_nomulti = y_test1_nomulti.astype('int')
    y_test2_nomulti = y_test2_nomulti.astype('int')
    y_test3_nomulti = y_test3_nomulti.astype('int')
    y_test4_nomulti = y_test4_nomulti.astype('int')
```

60-40

```
model1.fit(X_train1_nomulti, y_train1_nomulti)
model2.fit(X_train1_nomulti,y_train1_nomulti)
```

```
XGBClassifier

XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.1, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=8, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=100, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)
```

```
[ ] pred1_nomulti = model1.predict(X_test1_nomulti)
    pred2_nomulti = model2.predict(X_test1_nomulti)

    print('Model 1 XGboost Report %r' % (classification_report(y_test1_nomulti, pred1_nomulti)))
    print('Model 2 XGboost Report %r' % (classification_report(y_test1_nomulti, pred2_nomulti)))
```

```
Model 1 XGboost Report '              precision    recall  f1-score   support\n\n           0       0.62      0.72
Model 2 XGboost Report '              precision    recall  f1-score   support\n\n           0       0.62      0.63
```