ASDS 5302 – PRINCIPLES OF DATA SCIENCE

FINAL PROJECT REPORT

TEAM 1

HARIHARAN SELVAM (1002174644)

KESHAVI MANGALAPALLY (1002165375)

TABLE OF CONTENTS

SALARY PREDICTION

PROBLEM STATEMENT:

- Build an ML model to predict salary through the average salary with the help of other features.

DATASET:

- This dataset is taken from Kaggle data library.
- This is the data from the year 2018 from the Glassdoor job portal.
- This dataset includes only the data science related jobs, their locations, job description and many other aspects.

So, we've decided to solve this problem with the help of four stages.

1.Data Preprocessing

2. Descriptive Data Analysis

3.Exploratory Data Analysis

4.Explanation of features and the target.

5. Model Building

All the above-mentioned stages are briefly explained below.

INTRODUCING THE LIBRARIES:

The packages used in this problem are shown below.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score , r2_score
from scipy.stats import zscore
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
import matplotlib.pyplot as plt
from sklearn.linear_model import Lasso
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.linear_model import ElasticNet
from sklearn.multioutput import MultiOutputRegressor
import warnings
warnings.filterwarnings('ignore')
```

INFORMATION ABOUT THE DATASET:

- The original dataset has 742 rows and 28 columns.
- Here is the information of the dataset.

```
df_salary.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 742 entries, 0 to 741
Data columns (total 28 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Job Title           742 non-null    object 
 1   Salary Estimate     742 non-null    object 
 2   Job Description     742 non-null    object 
 3   Rating              742 non-null    float64
 4   Company Name        742 non-null    object 
 5   Location            742 non-null    object 
 6   Headquarters        742 non-null    object 
 7   Size                742 non-null    object 
 8   Founded             742 non-null    int64  
 9   Type of ownership   742 non-null    object 
 10  Industry            742 non-null    object 
 11  Sector              742 non-null    object 
 12  Revenue             742 non-null    object 
 13  Competitors         742 non-null    object 
 14  hourly              742 non-null    int64  
 15  employer_provided   742 non-null    int64  
 16  min_salary          742 non-null    int64  
 17  max_salary          742 non-null    int64  
 18  avg_salary          742 non-null    float64
 19  company_txt         742 non-null    object 
 20  job_state           742 non-null    object 
 21  same_state          742 non-null    int64  
 22  age                 742 non-null    int64  
 23  python_yn           742 non-null    int64  
 24  R_yn                742 non-null    int64  
 25  spark               742 non-null    int64  
 26  aws                 742 non-null    int64  
 27  excel               742 non-null    int64  
dtypes: float64(2), int64(12), object(14)
memory usage: 162.4+ KB
```

- The columns of this dataset are listed below.

```
df_salary.columns
```

```
Index(['Job Title', 'Salary Estimate', 'Job Description', 'Rating',
       'Company Name', 'Location', 'Headquarters', 'Size', 'Founded',
       'Type of ownership', 'Industry', 'Sector', 'Revenue', 'Competitors',
       'hourly', 'employer_provided', 'min_salary', 'max_salary', 'avg_salary',
       'company_txt', 'job_state', 'same_state', 'age', 'python_yn', 'R_yn',
       'spark', 'aws', 'excel'],
      dtype='object')
```

DATA PREPROCESSING:

It is the process of converting a raw data into a cleaned dataset.

The operations that have been performed during this data preprocessing are:

- Dropping unnecessary columns.
- Finding Duplicated values and removing it.
- Unique values of features and target.
- Information about unknown and other values

The step-by-step explanation of all these processes are given below.

Dropping Unnecessary columns:

- Here we've found columns like 'Sector', 'Location',' age', 'hourly', 'same_state', 'company_txt', 'Competitors', 'Founded', 'employer_provided', 'Headquarters' are repetitive and unnecessary.
- So, our columns count dropped from 28 to 18.
- To reduce the number of columns, we removed all of these columns from our dataset.

```
unnecessary_columns = ['Sector', 'Location','age','hourly', 'same_state', 'company_txt','Competitors',
                       'Founded','employer_provided','Headquarters']
df_salary_new = df_salary.drop(unnecessary_columns, axis=1, errors='ignore')
```

Finding Duplicated values and removing it:

- First, we searched for missing values and we haven't found any.
- Then we moved to find duplicated values.
- So we found 276 out of 742 columns are duplicated.
- Then we removed all those duplicated values by keeping the first found one on dataset and removed other duplicates.
- By doing this , we've reduced the dimensionality of our dataset to 466 rows and 18 columns.

Unique values of all columns:

- The unique values of each column of our dataset is given below.

```
df_salary_new.nunique()
```

```
Job Title            264
Salary Estimate      416
Job Description      463
Rating                31
Company Name         343
Size                   9
Type of ownership     11
Industry              60
Revenue               14
min_salary           119
max_salary           163
avg_salary           225
job_state             38
python_yn              2
R_yn                   2
spark                  2
aws                    2
excel                  2
dtype: int64
```

- And we've mentioned each and every unique value in our code file.

Information about unknown values:

The unknown values we found in our dataset are:

- revenue => -1, unknown/Non-Applicable
- industry => -1
- Type of ownership => -1, unknown
- size => unknown, -1
- Rating => -1.0

The next step is to remove all these values from our dataset.

```
remove_val = ['unknown','Unknown' 'Non-Applicable', -1, -1.0]
columns = ['Revenue', 'Industry', 'Type of ownership', 'Size', 'Rating']
df_salary_new[columns] = df_salary_new[columns].replace(remove_val, np.nan)
df_salary_new.dropna()
```

- And we've removed all these values through the above code.
- Then we checked for missing values once again.

```
df_salary_new.isnull().sum()
```

```
Job Title             0
Salary Estimate       0
Job Description       0
Rating               11
Company Name          0
Size                  0
Type of ownership     0
Industry              0
Revenue               0
min_salary            0
max_salary            0
avg_salary            0
job_state             0
python_yn             0
R_yn                  0
spark                 0
aws                   0
excel                 0
dtype: int64
```

- Through that we found 11 missing values in 'Rating' column and then we removed it.

```
df_salary_new = df_salary_new.dropna()
df_salary_new.isnull().sum()
```

```
Job Title            0
Salary Estimate      0
Job Description      0
Rating               0
Company Name         0
Size                 0
Type of ownership    0
Industry             0
Revenue              0
min_salary           0
max_salary           0
avg_salary           0
job_state            0
python_yn            0
R_yn                 0
spark                0
aws                  0
excel                0
dtype: int64
```

DESCRIPTIVE DATA ANALYSIS:

- Descriptive analytics is a statistical interpretation used to analyze historical data to identify patterns and relationships.
- At the start, the shape of our dataset is 731 rows and 18 columns.
- Here is the statistical Parameters.

```
df_salary_new.describe()
```

	Rating	min_salary	max_salary	avg_salary	python_yn	R_yn	spark	aws	excel
count	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000
mean	3.688372	73.396717	126.785226	100.090971	0.530780	0.002736	0.228454	0.238030	0.521204
std	0.570353	31.252418	46.784544	38.537585	0.499393	0.052271	0.420124	0.426169	0.499892
min	1.900000	10.000000	16.000000	13.500000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	3.300000	52.000000	96.000000	73.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	3.700000	69.000000	124.000000	96.500000	1.000000	0.000000	0.000000	0.000000	1.000000
75%	4.000000	90.000000	155.000000	122.500000	1.000000	0.000000	0.000000	0.000000	1.000000
max	5.000000	202.000000	306.000000	254.000000	1.000000	1.000000	1.000000	1.000000	1.000000

- While looking through this we can see that in some of these features, max value is more than the double of mean. This shows that there might be possibility for outliers.

Outlier Detection and Removal:

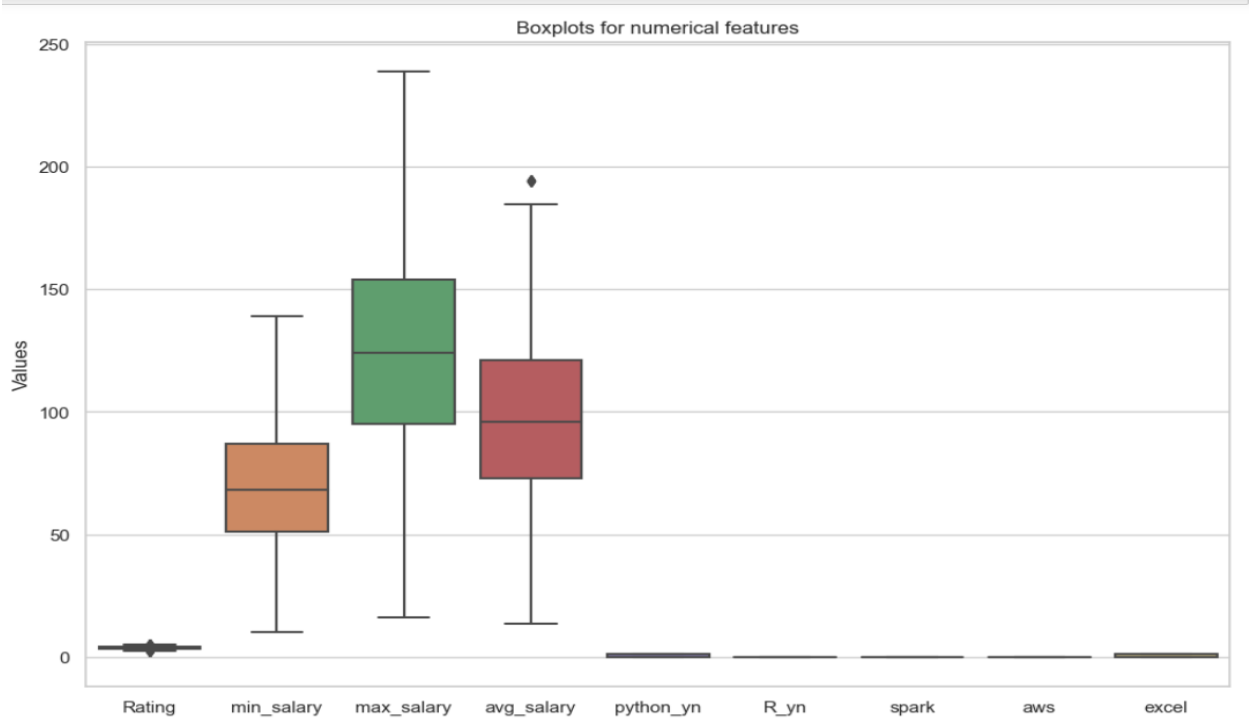
- We've tried to detect the outliers in numerical features and to remove it.

```
columns = [ 'Rating', 'min_salary', 'max_salary', 'avg_salary','python_yn','R_yn','spark','aws','excel']

def remove_outliers(columns):
    Q1 = df_salary_new[columns].quantile(0.25)
    Q3 = df_salary_new[columns].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = (df_salary_new[columns] < lower_bound) | (df_salary_new[columns] > upper_bound)
    df_salary_new[columns] = df_salary_new[columns][~outliers]

for column in columns:
    remove_outliers(column)
```

- After that, we've plotted the boxplot to verify it whether the above code worked or not.



- After removing the outliers, once again we checked the missing values.

```
df_salary_new.isnull().sum()
```

```
Job Title              0
Salary Estimate        0
Job Description        0
Rating                10
Company Name           0
Size                   0
Type of ownership      0
Industry               0
Revenue                0
min_salary            18
max_salary             6
avg_salary             9
job_state              0
python_yn              0
R_yn                   2
spark                167
aws                  174
excel                  0
dtype: int64
```

- And we found some missing values in our dataset. So, we removed it.

```
df_salary_new = df_salary_new.dropna()
```

```
df_salary_new.isnull().sum()
```

```
Job Title            0
Salary Estimate      0
Job Description      0
Rating               0
Company Name         0
Size                 0
Type of ownership    0
Industry             0
Revenue              0
min_salary           0
max_salary           0
avg_salary           0
job_state            0
python_yn            0
R_yn                 0
spark                0
aws                  0
excel                0
dtype: int64
```

- Now, the dimensionality of our dataset changed to 447 rows and 18 columns.

```
df_salary_new.describe()
```

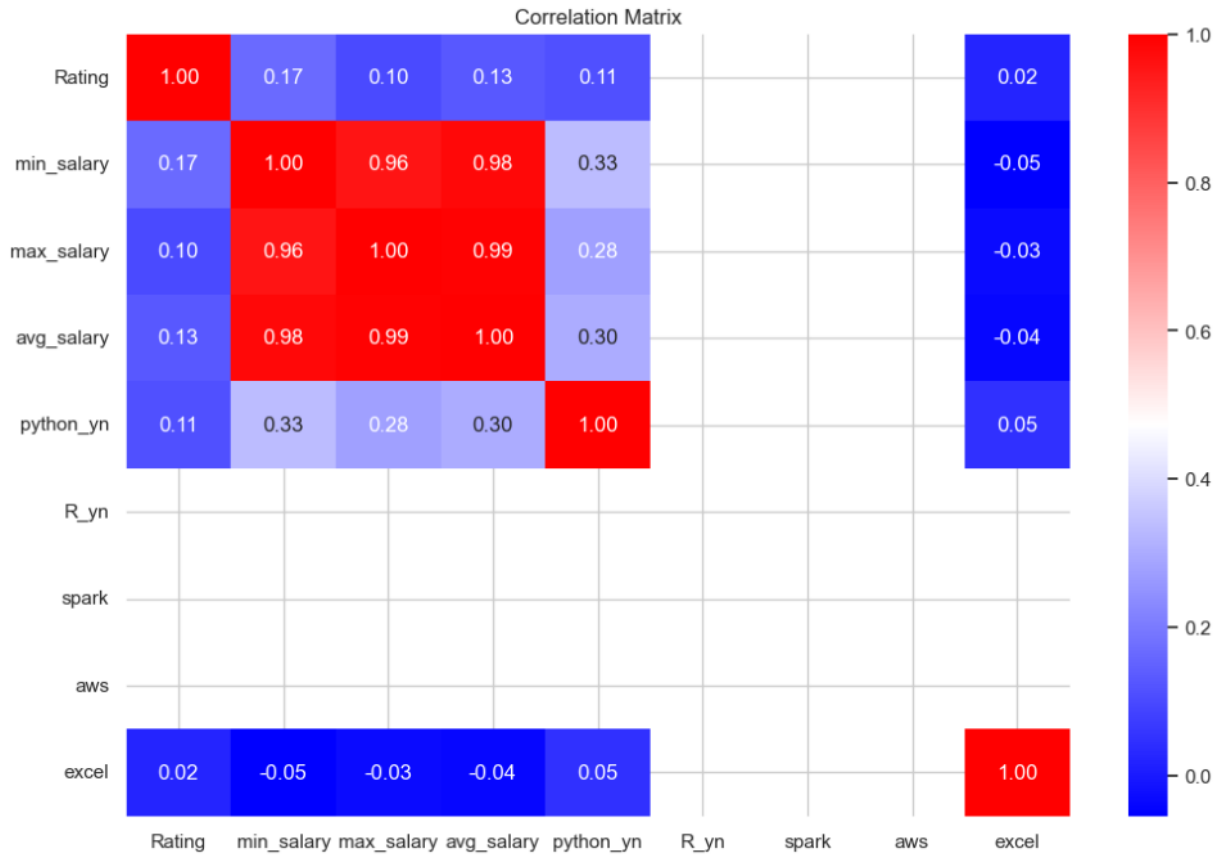
	Rating	min_salary	max_salary	avg_salary	python_yn	R_yn	spark	aws	excel
count	447.000000	447.000000	447.000000	447.000000	447.000000	447.0	447.0	447.0	447.000000
mean	3.624161	66.129754	116.923937	91.526846	0.389262	0.0	0.0	0.0	0.534676
std	0.561347	26.829409	43.634217	34.869172	0.488129	0.0	0.0	0.0	0.499355
min	2.300000	10.000000	16.000000	13.500000	0.000000	0.0	0.0	0.0	0.000000
25%	3.300000	45.000000	86.000000	66.250000	0.000000	0.0	0.0	0.0	0.000000
50%	3.700000	62.000000	113.000000	87.000000	0.000000	0.0	0.0	0.0	1.000000
75%	3.900000	84.000000	143.000000	114.500000	1.000000	0.0	0.0	0.0	1.000000
max	5.000000	139.000000	231.000000	180.000000	1.000000	0.0	0.0	0.0	1.000000

- And here is our final statistical parameters where everything looks sorted out.

EXPLORATORY DATA ANALYSIS:

- Exploratory Data Analysis (EDA) is a process of describing the data by means of statistical and visualization techniques in order to bring important aspects of that data into focus for further analysis.

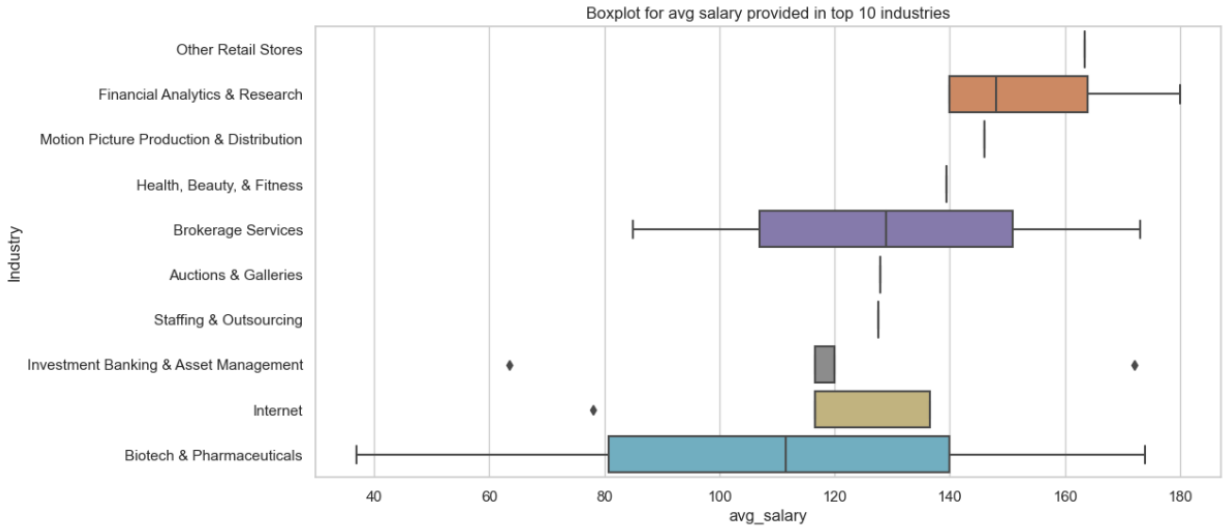
Correlation matrix:



- From this correlation matrix, we found that
- The correlation between max_salary and avg_salary remains the highest one (0.99)
- There is no relation between R_yn and spark, spark and aws as the correlation between them is 0.

BOXPLOT:

- Here we've plotted box plot to show the average salary provided by top 10 industries.
- This helped us to find top 10 industries and the salaries provided by them.



From this Boxplot,

Top 10 industries for Data Scientist interms of salaries

1.Other retail stores

2.Financial Analytics and research

3.Motion Picture Production & Distribution

4.Health, Beauty & Fitness

5.Brokerage services

6.Auctions and Galleries

7.Staffing and outsourcing

8.Investment Banking & Asset management

9.Internet

10. Biotech and pharmaceuticals

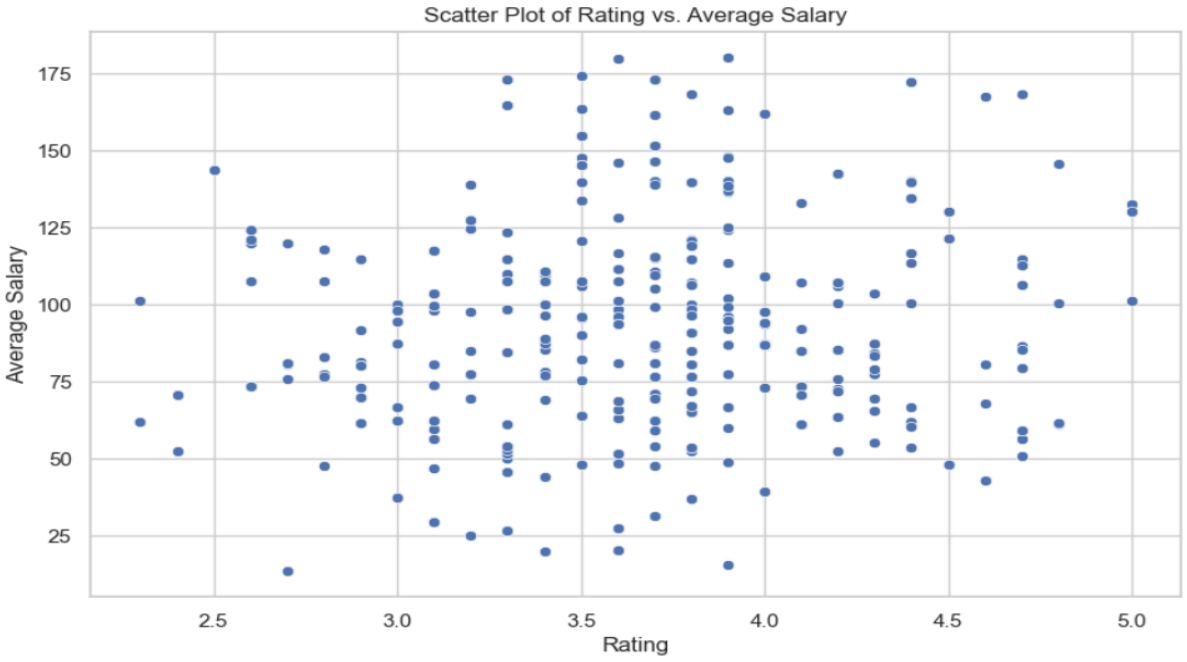
Brokerage services, Auctions and Galleries, Staffing and outsourcing have nearly same median.

Financial Analytics and research, Motion Picture Production & Distribution have nearly same median.

Biotech and pharmaceuticals have highest range of variance.

SCATTERPLOT:

- We've plotted scatterplot to show the influence of rating on average salary.

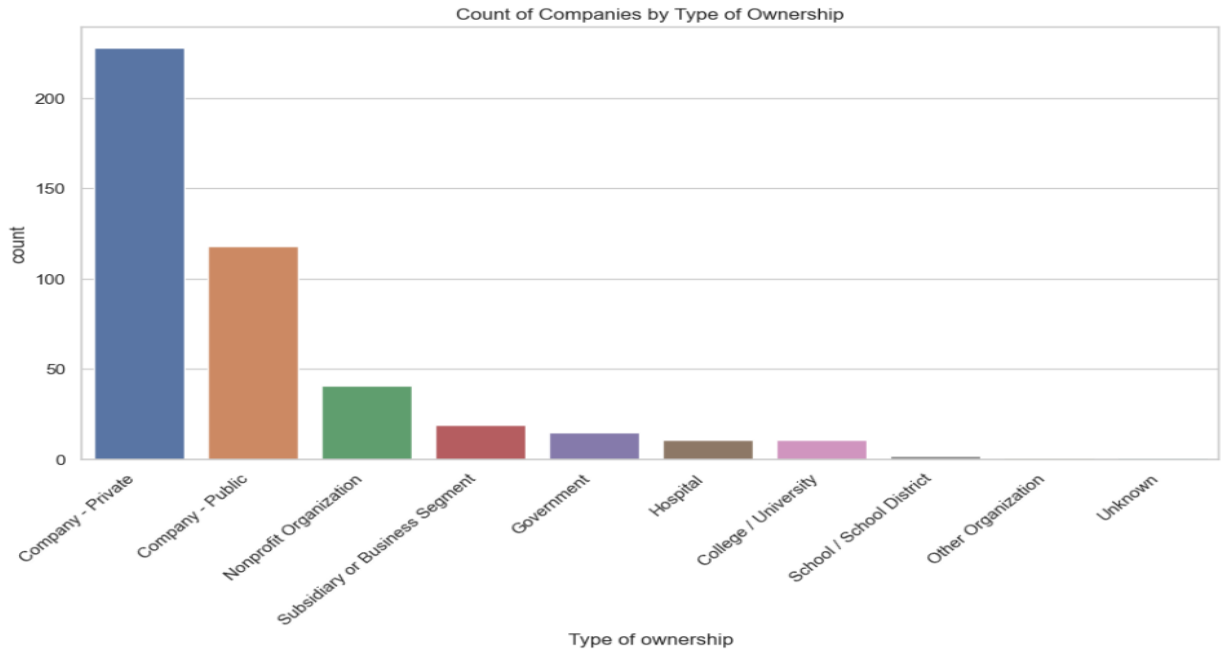


From this scatterplot,

- Highest salary is provided by the companies have rating of 3.5 to 4.0
- 3 positions fall under the highest rating of 5.0
- Most of the positions are offered by the companies having a rating of 3.5 to 4.0.

Count plot:

- We've used count plot to show the number of companies with the type of ownership.

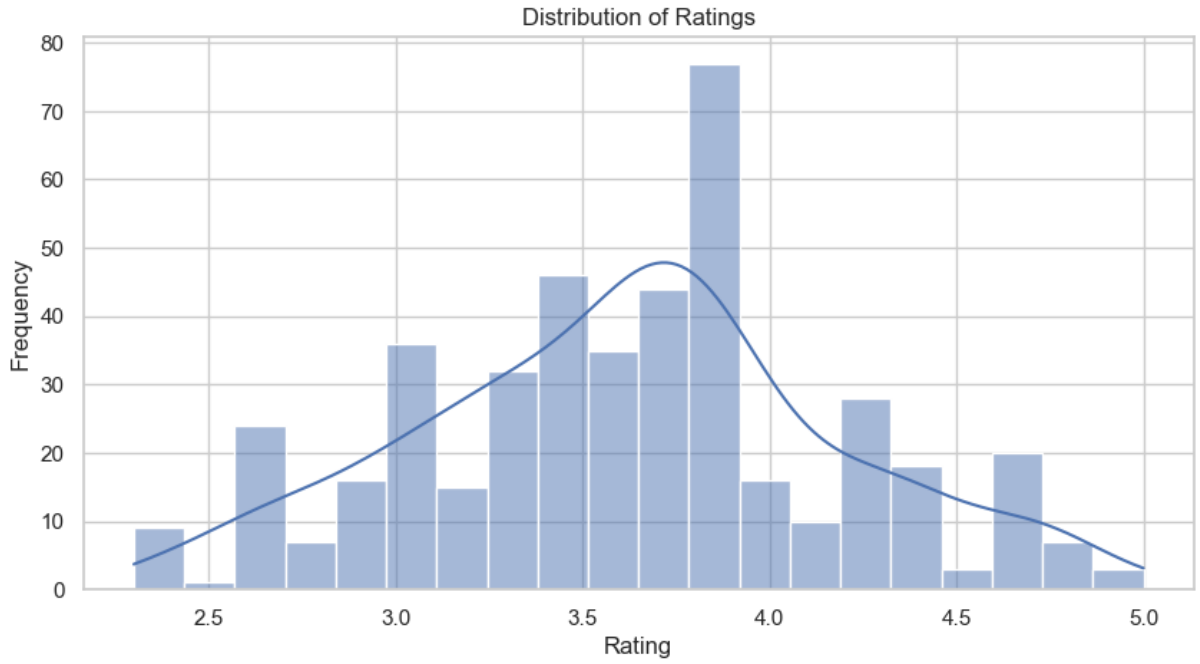


From this count plot,

- More than 200 positions were offered by the private companies.
- Around 100 to 150 positions were offered by public companies.
- Around 50 positions were offered by nonprofit organizations.
- Around 25 positions expected to be offered by Subsidiary or business segment.
- At least 15-20 positions were offered by the Government.
- Hospitals, college/university have same number of job positions.

Histogram:

- We've plotted histogram to show the flow of rating.



From this Histogram,

- Most of the companies have rating of 3.9
- Only around 5 companies expected to have highest rating of 5.0

LABEL ENCODING:

- Label Encoding has been performed to convert our categorical variables into numerical variables.
- As there are many categorical features and to make use of it, we are converting our categorical variables to numerical ones.

```
from sklearn.preprocessing import LabelEncoder
categorical_columns = ['Job Title', 'Size', 'Type of ownership', 'Revenue', 'job_state',
                       'Salary Estimate', 'Job Description', 'Company Name','Industry']

label_encoder = LabelEncoder()

for column in categorical_columns:
    column_name = column.strip()
    df_salary_new[column_name + '_encoded'] = label_encoder.fit_transform(df_salary_new[column_name].astype(str))

df_salary_new = df_salary_new.drop(categorical_columns, axis=1)
```

- We used the above code to perform Label Encoding.
- And here is the update of our new columns.

```
df_salary_new.columns
```

```
Index(['Rating', 'min_salary', 'max_salary', 'avg_salary', 'python_yn', 'R_yn',
       'spark', 'aws', 'excel', 'Job Title_encoded', 'Size_encoded',
       'Type of ownership_encoded', 'Revenue_encoded', 'job_state_encoded',
       'Salary Estimate_encoded', 'Job Description_encoded',
       'Company Name_encoded', 'Industry_encoded'],
      dtype='object')
```

EXPLANATION OF FEATURES AND TARGET:

- According to our problem statement,
- Here we are using ‘avg_salary’ as our target and 'max_salary','min_salary', 'Job Title_encoded' as features.

Train and Test:

- Our train and test ratio are given to 70:30.
- And we’ve reshaped the values of X_train, X_test to match the dimensionality

MODEL BUILDING:

This is the final stage of solving our problem statement.

Here we’ve used 4 different algorithms such as

1. Linear Regression
2. Decision Tree
3. Random Forest
4. Gradient Boosting

All these algorithms fall under Regression.

LINEAR REGRESSION:

- Linear regression is a fundamental machine learning algorithm used for predicting numerical values based on input features. It assumes a linear relationship between the features and the target variable. The model learns the coefficients that best fit the data and can make predictions for new inputs.
- The code which we used for Linear Regression is given below.

```
model = LinearRegression()
model.fit(X_train_array, y_train)
y_pred_lr = model.predict(X_test_array)
error = y_actual - y_pred_lr
absolute_error = abs(error)
mean_absolute_error = absolute_error.mean()

print("The error value is:\n", error)
print("\n Absolute error is\n" , absolute_error)
print("\n Mean absolute error is:\n", mean_absolute_error)


mse = round(mean_squared_error(y_test, y_pred_lr),2)
print("Mean Squared error:", mse)
print(f"Accuracy of model = {round(r2_score(y_test, y_pred_lr),4)*100}%")
```

- For Linear Regression we got an accuracy of 67.89%.

DECISION TREE REGRESSION:

- Decision trees is a type of supervised machine learning algorithm that is used by the Train Using Auto ML tool and classifies or regresses the data using true or false answers to certain questions.
- The code we used for decision tree is given below.

```
decision_tree_model = DecisionTreeRegressor(max_depth=5)
decision_tree_model.fit(X_train_array, y_train)
decision_tree_predictions = decision_tree_model.predict(X_test_array)
print(f'Decision Tree Regression MSE: {mean_squared_error(y_test, decision_tree_predictions)}')
print(f"Accuracy of model = {round(r2_score(y_test, decision_tree_predictions),4)*100}%")
```

```
Decision Tree Regression MSE: 764.7106043167869
Accuracy of model = 67.75%
```

- Here we got an accuracy of 67.75%.

RANDOM FOREST REGRESSION:

- Random forest regression is a supervised learning algorithm and bagging technique that uses an ensemble learning method for regression in machine learning.

```
random_forest_model = RandomForestRegressor(n_estimators=100)
random_forest_model.fit(X_train_array, y_train)
random_forest_predictions = random_forest_model.predict(X_test_array)
print(f'Random Forest Regression MSE: {mean_squared_error(y_test, random_forest_predictions)}')

print(f"Accuracy of model = {round(r2_score(y_test, random_forest_predictions),4)*100}%")
```

```
Random Forest Regression MSE: 703.2111351506074
Accuracy of model = 70.76%
```

- Here we got an accuracy of 70.76%

GRADIENT BOOSTING REGRESSION:

- Gradient boosting is a machine learning technique used in regression and classification tasks, among others. It gives a prediction model in the form of an ensemble of weak prediction models

```
X_train_np = X_train.to_numpy().reshape(-1, 1)
X_test_np = X_test.to_numpy().reshape(-1, 1)
y_train_np = y_train.to_numpy()


gradient_boosting_model = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1)
multi_output_model = MultiOutputRegressor(gradient_boosting_model)
multi_output_model.fit(X_train_np, y_train_np)
gradient_boosting_predictions = multi_output_model.predict(X_test_np)
gradient_boosting_mse = mean_squared_error(y_test, gradient_boosting_predictions)


print(f'Gradient Boosting Regression MSE: {gradient_boosting_mse}')
print(f"Accuracy of model = {round(r2_score(y_test, gradient_boosting_predictions),4)*100}%")
```

```
Gradient Boosting Regression MSE: 668.3832333296408
Accuracy of model = 72.17%
```

- Here we got an accuracy of 72.17%

Conclusion:

- Gradient Boosting performed well among others. And we've built a salary prediction model based on average salary with the help of job title, minimum salary and average salary. As this is just a predictive model, we can't expect full accuracy.

References:

Dataset link - https://www.kaggle.com/datasets/thedevastator/jobs-dataset-from-glassdoor