Exploring the Data Job Market: An Analysis and Prediction of Job Listings and Salary Trends

Milestone: Draft of Final Project Report

Group 7
Valli Meenaa Vellaiyan
Hrithik Sarda

857-832-0123
978-654-0445

vellaiyan.v@northeastern.edu
sarda.h@northeastern.edu

Percentage of effort contributed by student 1: _____50_____

Percentage of effort contributed by student 2: _____50_____

Signature of student 1: _____Valli Meenaa_____

Signature of student 2: _____Hrithik Sarda_____

Submission date: _____21st April 2023_____

Table of Contents

PROBLEM SETTING

The problem setting is as follows: analyze and understand the current job market for Data Scientists, Data Engineers, and Data Analysts, such as identifying:

- The most in-demand skills and qualifications
- The most common industries hiring data scientists/engineers/analysts etc.
- The regions or cities with the highest concentration of data job openings
- The average salary range for various positions
- The most common job titles
- The most common words and phrases used in job descriptions to identify specific tasks and responsibilities that are most sought after by employers.
- Most importantly, to use the job descriptions to develop a model that can predict the job's salary range, given the job description.
- Additionally, it can also be used to build a model that can predict the job industry, given the job description.

A variety of techniques such as data cleaning, data visualization, statistical analysis, and machine learning will be used to understand and provide necessary insights from the dataset.

PROBLEM DEFINITION

The project's goal is to solve the issue of unreported salaries for data scientists, data analysts, and data engineers. Companies can also use this methodology to determine the appropriate starting compensation for new hires. A model is developed based on various criteria to group firms for staff transfers. The projected pay from the model might help employees choose the next ideal position. The project uses job descriptions as input to forecast job attributes such as income range, industry, job title, and other aspects of the work to study and understand the employment market for data scientists, engineers, and analysts. In summary, this model uses machine learning techniques to analyze and understand the job market for data scientists/analysts/engineers by predicting prospective job characteristics.

DATA SOURCES

The datasets for “Data Analyst Jobs”, “Data Scientist Jobs”, and “Data Engineer Jobs” have been taken from https://www.kaggle.com/, an open-source, secure online community, which allows users to browse through numerous datasets. The links for the three datasets are provided below:
i) https://www.kaggle.com/datasets/durgeshrao9993/data-analyst-jobs-datset
ii) https://www.kaggle.com/datasets/andrewmvd/data-scientist-jobs
iii) https://www.kaggle.com/datasets/andrewmvd/data-engineer-jobs

Data Description

After combining the three datasets, the final dataset contains 16 columns (15 attributes and 1 target variable - “Salary Estimate” and 8676 rows. The dataset contains job listings for data scientist roles, data analyst roles, and data engineer roles. It includes several fields such as job title, company name, location, salary, and job description. All the attributes and their descriptions are given below:

No.	Attribute	Description
1.	ID	To identify each record uniquely
2.	Job Title	The title of the job listing
3.	Salary Estimate	The salary range for the job, and this is our response variable
4.	Job Description	A text description of the job responsibilities and qualifications
5.	Rating	Job rating out of 5
6.	Company Name	The name of the company that posted the job listing
7.	Location	The location of the job
8.	Headquarters	Headquarters of the company providing the respective job role
9.	Size	Number of employees working at the company currently
10.	Founded	The year in which the company was founded
11.	Type of Ownership	Type of business [Public, Private, Sports, etc.]
12.	Industry	The industry in which the company operates
13.	Sector	The sector to which the industry belongs to
14.	Revenue	Total income of the company.
15.	Competitors	Other companies that are current competitors to the listed company
16.	Easy Apply	Tells if it is easy to apply or requires specific reference/connections

Table 1: Initial Column Names with their Descriptions

Data Collection (integration of all the three datasets)

To begin our analysis, we initially reviewed the three datasets (pertaining to data analysts, data scientists, and data engineers) independently, with the aim of exploring the distinct columns of each dataset individually. Upon analyzing the three datasets, we identified an extraneous column in the Data Analysts dataframe that we dropped, as well as two unnecessary columns in the Data Scientists dataframe that were also removed. However, we did not find any such columns in the Data Engineers dataset.

Next, we merged the three individual dataframes into a consolidated dataframe named "df_comb". Following this, we removed any duplicated rows from the merged dataframe, resulting in the elimination of 16 rows. Ultimately, our final dataframe contained 8674 rows.

DATA EXPLORATION

Data Cleaning

I. Dropping columns based on number of null values:

To begin with, we computed the number of missing values in each column of the combined dataset. Next, we visualized this data using a horizontal bar chart (Fig 1). The chart allows for an easy comparison of the number of missing values across the different columns.

The column with the highest number of null values is "Easy Apply", with 8286 null values (95.52%). This indicates that a vast majority of the job postings in the combined dataset do not have an easy apply option. Hence, we remove that column. Additionally, we chose to drop the "Competitors" column as well, as it had 71.84% null values (6232). The decision to remove the columns was made based on the understanding that null values may hinder our ability to analyze the data in the columns, and we could obtain useful insights without those columns. We also remove the Industry column, since it is just a subset of Sector (repetitive).

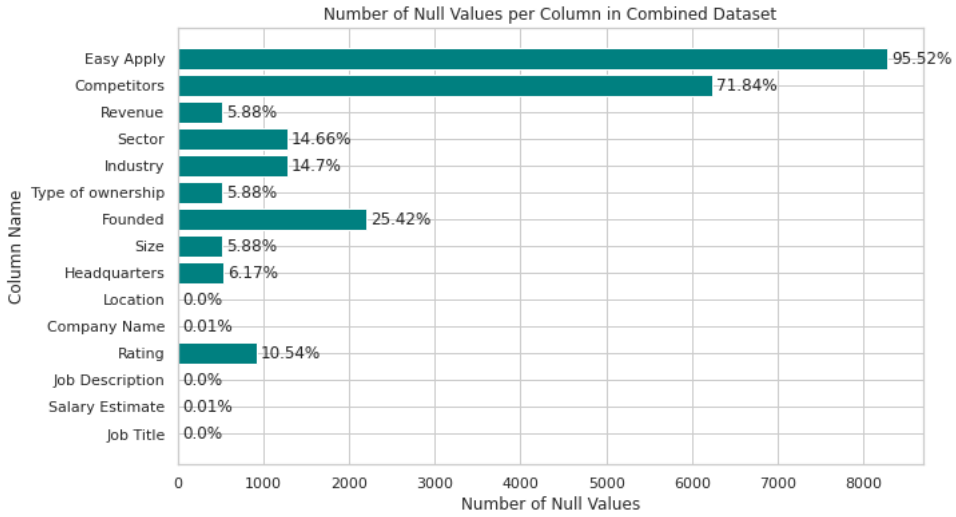
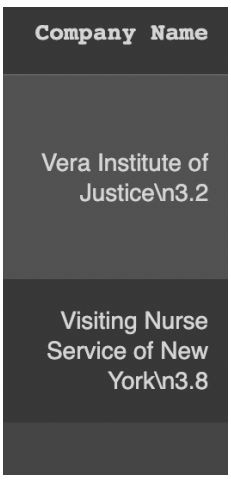


Fig 1: Horizontal Bar Chart to represent number of null values in each column

II. Cleaning “Company Name”:

Moving on, we made use of various data-cleaning techniques to ensure consistency and reduce redundancy in the columns. In the next step of our data cleaning process, we focused on cleaning the "Company Name" column. Fig 2 represents the state of the "Company Name" column before cleaning, while Fig 3 depicts the state of the column after cleaning.



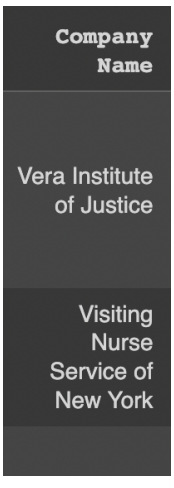
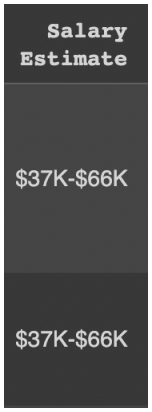


Fig 2: Company Name before cleaning

Fig 3: Company Name after cleaning

III. Cleaning “Salary Estimate”:

We proceeded to split the "Salary Estimate" column into two separate columns, namely "Starting_Salary" and "Ending_Salary". We also cleaned the column by removing the dollar symbol and "K", multiplying the numerical values by 1000, and converting the values to float type for ease of future analysis. We noticed that in a dataset with nearly 9000 data points, there were only approximately 111 unique values for the starting and ending salaries. To address this, we added a bit of noise, or jitter, to the salary values to improve the diversity of the data. To accomplish this, we utilized the numpy.random.normal() function and generated random numbers from a normal distribution, adding the result to each salary value. The amount of jitter added was equivalent to 10% of the standard deviation of the salary data. By performing these steps, we were able to prepare the salary data for further analysis.



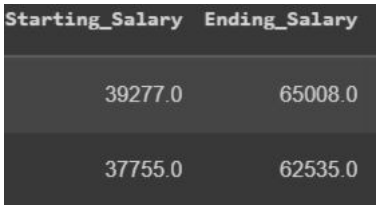


Fig 4: Salary Estimate before cleaning

Fig 5: Salary Estimate after cleaning

Therefore, we have two new Variables: 1) Starting_Salary 2) Ending_Salary. We combine these two columns into a single variable called **“average_salary”**, which is our **response/target variable**.

V. Cleaning “Revenue” Column:

Now, we label encode the revenue column such that we classify the revenues into 4 categories: “Low”, “Medium”, “High”, and “Very high”.

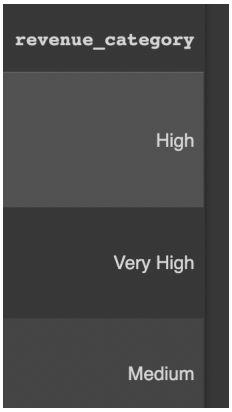


Fig 8: Revenue after cleani

VI. Creating a "job_state" column from "Location":

We split the "Location" column on "," and extract the state from it, and assign it to a new column called "job_state".

VII. Finding Company's Age from "Founded" column:

We perform a small mathematical calculation to compute the company's age, based on the year provided in the "Founded" column.

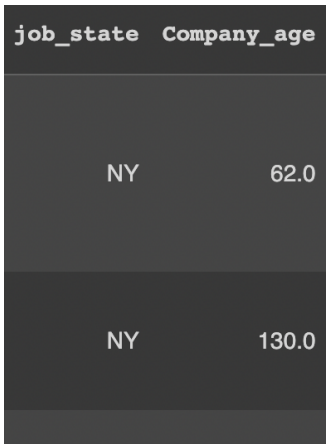


Fig 10: job_state and company_age columns

VIII. Working on the "Job Title" column:

Since "Job Title" is a categorical variable, we will be required to one-hot encode it, in order to be able to use them as predictors for our final model. However, this column has a very huge bracket of roles. Therefore, we first write a loop to classify the job titles into just nine classes and add them as additional features to our dataset (data engineer, analyst, data scientist, machine learning engineer, AI engineer, cloud engineer, manager, director, and others). We also create another column called "seniority" based on the "Job Title" column, and we obtain two classes for the same (senior or junior). We further convert the seniority column into an ordinal categorical variable by doing label encoding.

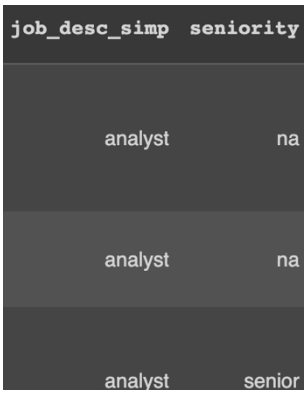


Fig 11: job_desc_simp and seniority columns

X. Working on the "Ownership" column:

This column also requires us to do one-hot encoding to convert the categorical variables into additional features based on the number of unique values in this categorical feature. "Ownership" is converted into five different columns (private, public, educational_institution, government, and other organizations).

```
df_comb['Type of ownership']

0       other organisations
1       other organisations
2                   private
3       other organisations
4                   private
                ...
8159                private
8160                private
8161                private
8162                 public
```

Fig 13: modified type of ownership column

XI. Adding a "Seniority" column:

We create a function called seniority to extract "senior" and "junior" from the "Job Title" column. Further, we ranked them as 0, 1, 2.

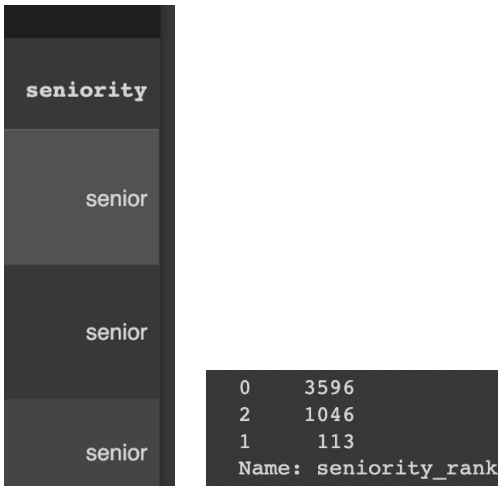


Fig 14: modified "seniority" column

XII. Preprocessing "Company Size" column:

We label-encoded the size column such that we have three categories of sizes – "small", "medium", and "large".

```
Large     2050
Small     1355
Medium    1350
Name: size_category, dtype: int64
```

Fig 15: label-encoding the "company size" column

Data Visualization

We've visualized the starting salaries and ending salaries through box plots. By creating box plots for the starting and ending salaries, we were able to gain insights into the salary ranges and how they vary across the different job titles in our dataset. The distribution is mostly uniform, with the exception of a couple of outliers.

A distplot of the "Rating" column would give a graphical representation of the distribution of the "Rating" values in the dataset. It can be observed that the ratings of 1-5 are spread over all the different job postings in an almost normally distributed manner. The slightly left-skewed distribution implies that the median rating is likely higher than the mean rating. This can happen when there are a few companies with very high ratings that pull up the mean but not the median. The skewness can also indicate that there are more companies with lower ratings in the dataset than those with higher ratings. The outlier at rating 5 suggests that there may be some companies that are exceptional in terms of their ratings.

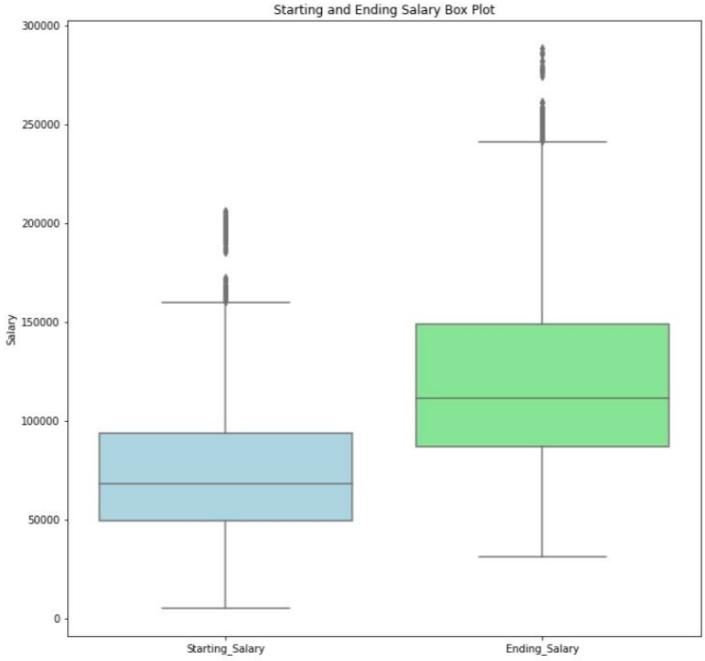


Fig 16: Box Plot visualization of Starting and Ending salaries

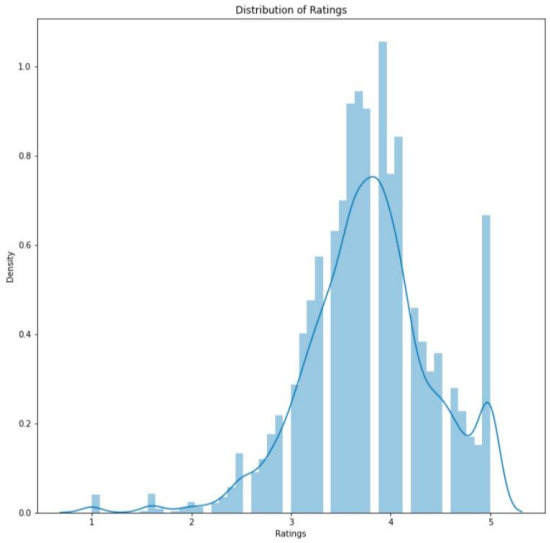


Fig 17: Distplot of Ratings

Next, we've taken the mean salary, average size of the company, rating, and average revenue generated by the company each year to create a pairplot, as these are the only numerical columns. We cannot observe any correlation between these variables as we have many overlapping values.

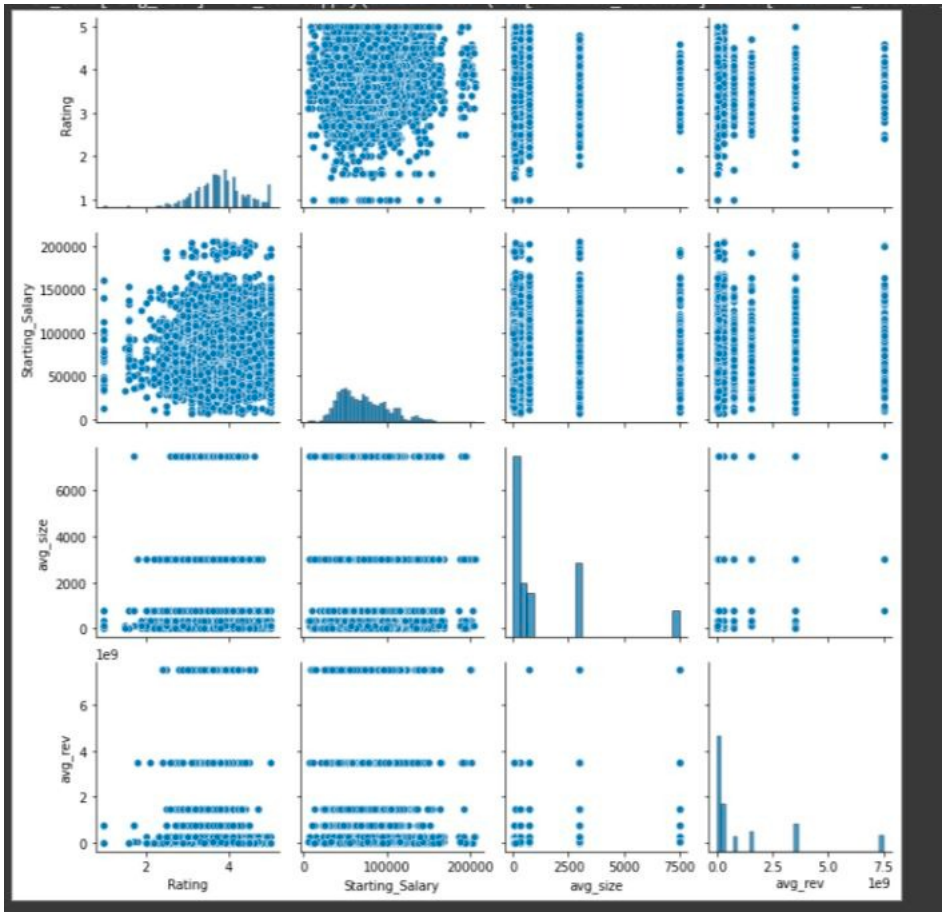


Fig 18: Pairplot among all numerical variables

We've also created distplots for the minimum revenue and company age columns for filling the missing values:

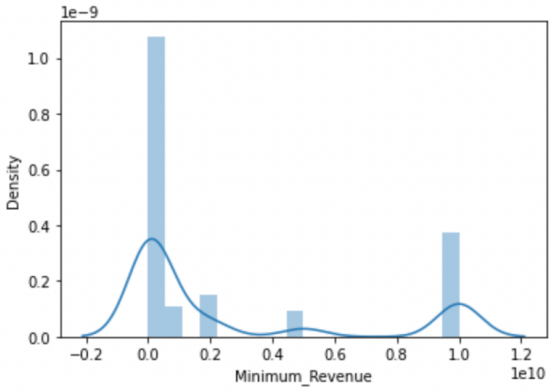


Fig 19: Distplot of minimum revenue

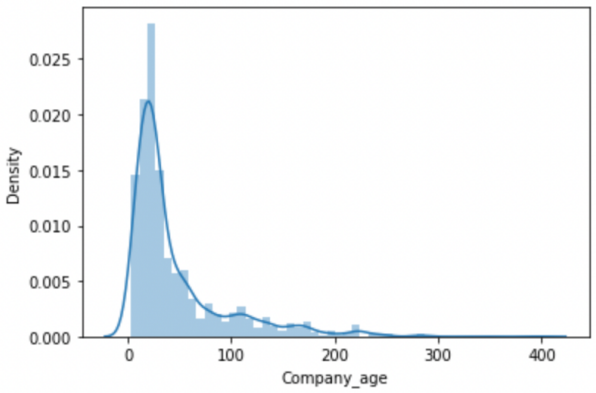


Fig 20: Displot of company age

We have three categories of jobs in our dataset. The below chart shows the names of the top 20 companies that provide these roles in bulk.

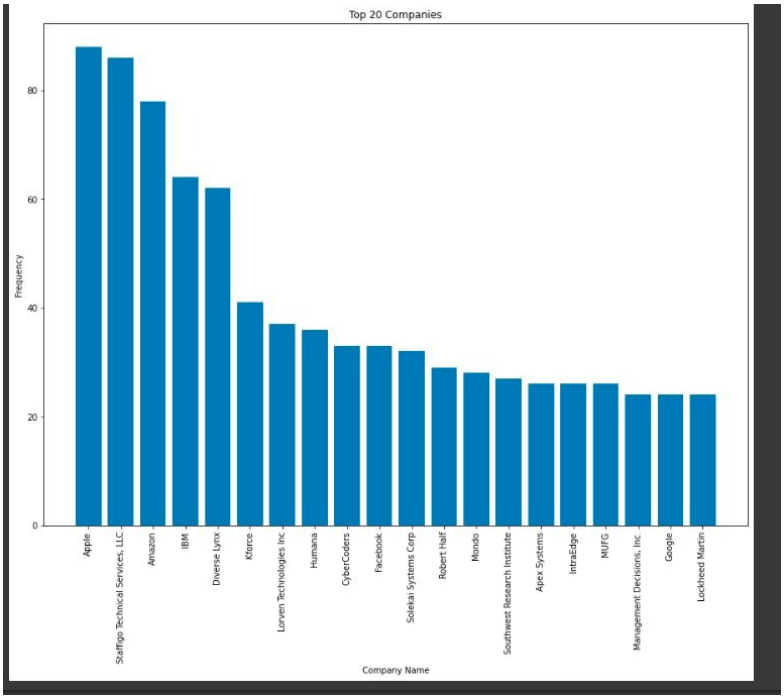


Fig 21: Bar plot for top 20 companies offering DA, DE, and DS roles

Data Processing

I. Sector vs. Mean Salary

We can use a statistical measure called Cramer's V, which is a measure of association between two nominal variables (mean salary and Sector). It ranges from 0 to 1, where 0 indicates no association and 1 indicates a perfect association. After performing a Cramer's V test on the "Sector" column and the mean salary, we obtained a low value of 0.0021. This value suggests that there is no significant association between the Sector that a specific company/job belongs to and our response variables. This means that the salary is independent of the sector and vice-versa.

```
import researchpy as rp
import pandas as pd
df_comb["mean_salary"] = df_comb.apply(lambda row: (row["Starting_Salary"] + row["Ending_Salary"])
crosstab, test_results = rp.crosstab(df_comb['Sector'], df_comb['mean_salary'], test='chi-square')
cramers_v = np.sqrt(test_results.iloc[2,1] / (df_comb.shape[0] * (min(crosstab.shape) - 1)))
print(f"Cramer's V: {cramers_v}")
```

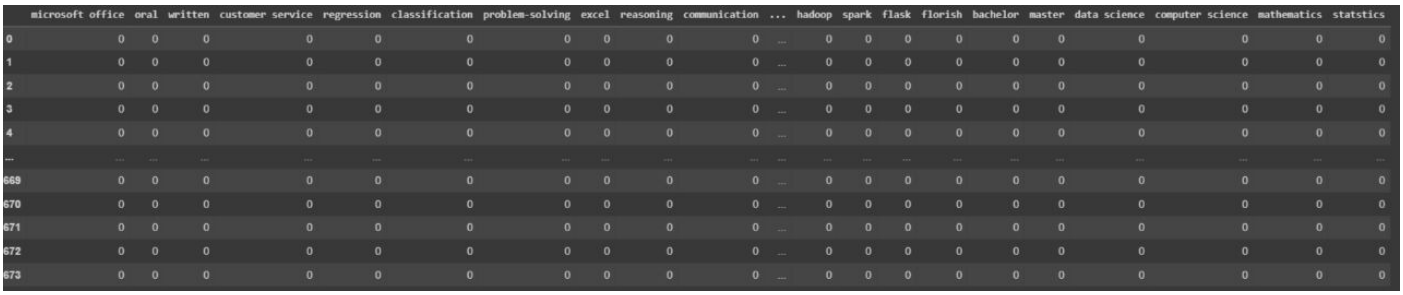
```
Cramer's V: 0.002140338012218702
```

Fig 22: Cramer's V Test on Sector and Mean Salary

II. Processing "Job Description"

We'll be performing Natural Language Processing (NLP) on the Job Description column. The job description column consists of a string of words in the form of a paragraph. Each paragraph contains information about the job posting, required qualifications, and skills. We merged the entire column of Job Descriptions into one big text chunk and found the frequency of words using NLP and regular expressions. This involves preprocessing the text data, tokenizing the text, converting it into a numerical representation using count vectorization or TF-IDF, and analyzing the frequency of words. We chose 200 such words and then manually

filtered 60 skills ['microsoft office', 'oral', 'written', 'customer service', 'regression', 'classification', 'problem-solving','excel', 'reasoning', 'communication', 'charting', 'python', 'sql', 'management', 'creative', 'dashboards', 'machine learning', 'data visualisations', 'tableau', 'powerbi', 'r', 'etl', 'extract/transform/load', 'nlp', 'nltk', 'natural language processing', 'aws', 'predictive modelling', 'computer vision', 'cloud computing', 'software', 'azure', 'gcp', 'distributed system', 'data reporting', 'cleaning', 'modelling', 'big data', 'deep learning', 'neural network', 'statastical', 'analysis','modeling', 'databases', 'mongodb', 'network', 'cypher', 'hadoop', 'analytics', 'hadoop', 'spark', 'flask', 'florish', 'bachelor', 'master', 'data science', 'computer science', 'mathematics', 'statstics']. We then stored the skills in a list and created a data frame with these skills as columns. Now, we go over each row in our original dataframe and check if each job description contains any of the skills, and if it does, then we assign 1 to a particular skill, else 0. Finally, we merge the skills dataframe with our "df_comb".

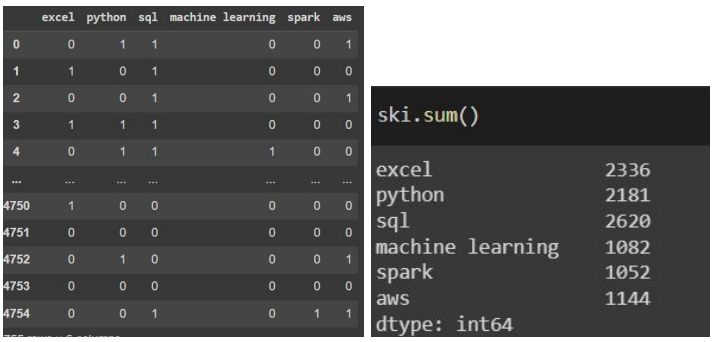


	microsoft office	oral	written	customer service	regression	classification	problem-solving	excel	reasoning	communication	...	hadoop	spark	flask	florish	bachelor	master	data science	computer science	mathematics	statstics
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
669	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
670	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
671	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
672	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
673	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

Fig 23: Pivot form of skills

The number of years of experience in each job description is present in various formats, we're working on its extraction.

Since we got way too many columns, we reduced the number if skills to 6: ['excel', 'python', 'sql', 'machine learning', 'spark', 'aws'].



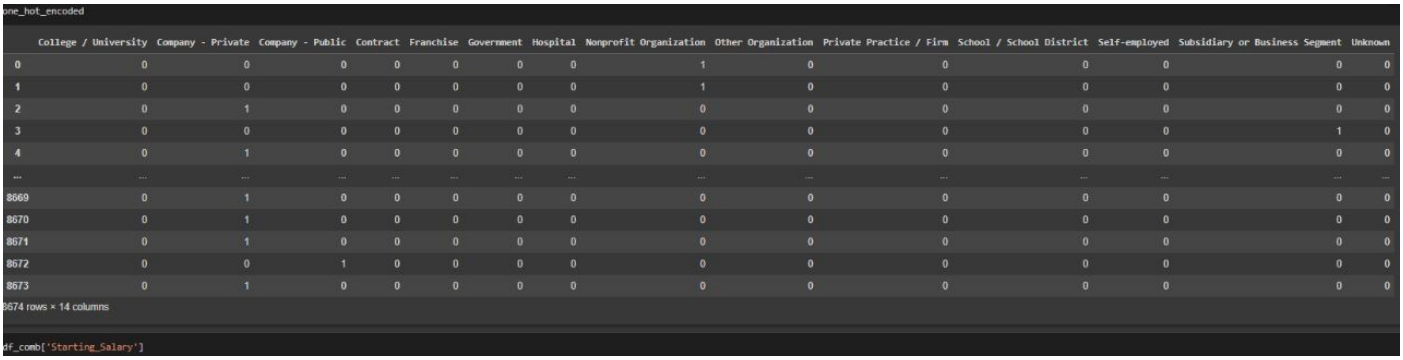
	excel	python	sql	machine learning	spark	aws
0	0	1	1	0	0	1
1	1	0	1	0	0	0
2	0	0	1	0	0	1
3	1	1	1	0	0	0
4	0	1	1	1	0	0
...	...	...	...	...	...	...
4750	1	0	0	0	0	0
4751	0	0	0	0	0	0
4752	0	1	0	0	0	1
4753	0	0	0	0	0	0
4754	0	0	1	0	1	1

```
ski.sum()

excel              2336
python             2181
sql                2620
machine learning   1082
spark              1052
aws                1144
dtype: int64
```

Fig 24: Final 6 chosen skills

III. One-hot encoding of "Type of Ownership" column:

We have converted this categorical variable into a numerical variable using one-hot encoding. We will using the long pivot form of the dataframe for predicting salaries.



one_hot_encoded

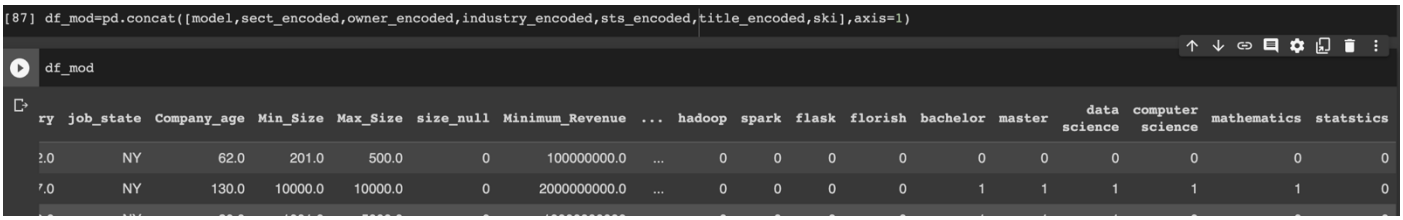
	College / University	Company - Private	Company - Public	Contract	Franchise	Government	Hospital	Nonprofit Organization	Other Organization	Private Practice / Firm	School / School District	Self-employed	Subsidiary or Business Segment	Unknown
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	1	0
4	0	1	0	0	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8669	0	1	0	0	0	0	0	0	0	0	0	0	0	0
8670	0	1	0	0	0	0	0	0	0	0	0	0	0	0
8671	0	1	0	0	0	0	0	0	0	0	0	0	0	0
8672	0	0	1	0	0	0	0	0	0	0	0	0	0	0
8673	0	1	0	0	0	0	0	0	0	0	0	0	0	0

8674 rows × 14 columns

df_comb['Starting_Salary']

Fig 25: One-hot-encoded form of "Type of Ownership" column

Similarly, we've dine one-hot encoding for 'Sector', 'Type of Ownership', 'Industry', 'job_state', and 'job_desc_simp' columns.



```
[87] df_mod=pd.concat([model,sect_encoded,owner_encoded,industry_encoded,sts_encoded,title_encoded,ski],axis=1)
```

df_mod

ry	job_state	Company_age	Min_Size	Max_Size	size_null	Minimum_Revenue	...	hadoop	spark	flask	florish	bachelor	master	data science	computer science	mathematics	statstics
2.0	NY	62.0	201.0	500.0	0	100000000.0	...	0	0	0	0	0	0	0	0	0	0
7.0	NY	130.0	10000.0	10000.0	0	2000000000.0	...	0	0	0	0	1	1	1	1	1	0

Fig 26: Final One-hot-encoded form

Dimension Reduction and Variable Selection

- The original dataset had several columns, including ID, Job Title, Salary Estimate, Job Description, Rating, Company Name, Location, Headquarters, Size, Founded, Type of Ownership, Industry, Sector, Revenue, Competitors, and Easy Apply.
- The "Easy Apply" column was removed due to having 95.5% of Null values, and the "Competitors" column was removed due to having 71.1% of Null values.
- Variable selection was performed to reduce the number of features in the dataset. The "Industry" column was removed as it had similar content to the "Sector" column.
- Certain columns were one-hot encoded to transform categorical data into numerical data, and then the original categorical columns were removed.
- PCA analysis was performed to further reduce the dimensionality of the dataset. The explained variance and distribution of classes were analyzed using PC1 and PC2.

PREDICTION MODELLING

Data Partitioning

After finishing the process of feature engineering (data cleaning, processing, manipulation, and transformation), we use MinMaxScaler and fit_transform to scale the required columns. Next, we split the dataset into training and test sets in the ratio of 75:25. This implies that 75% of the dataset is used for training the prediction models and the remaining 25% of the dataset is used as the test set for evaluation of the prediction performance of those models. X_train contains 6123 records and 149 variables while X_test

contains 2041 records and 149 variables respectively. Y_train contains 6123 records and 1 variables while Y_test contains 2041 records and 1 variable respectively.

Data Mining Models

Our response/dependent variable (average_salary) is a continuous, numeric variable. Therefore, we initially considered our project to be a regression problem.

1. Linear regression

Linear regression is a statistical technique used to create a model that shows the relationship between a dependent variable and one or more independent variables. In the case of salary prediction, the objective is to estimate the "average_salary" of an employee by considering the independent variables discussed earlier. The effectiveness of our linear regression model in generating accurate predictions depends on the quality and significance of the data used to construct the model.

Advantages:

a) One of the main advantages of using linear regression is that it is a relatively simple and straightforward statistical method to implement.
b) The output coefficients of the model are easy to interpret, making it easier to understand the relationship between the independent and dependent variables.
c) Linear regression is also ideal when there is a clear linear relationship between the dependent and independent variables, as it has less model complexity.

Disadvantages:

a) In some cases, a significant amount of feature engineering may be required to ensure that the model performs well.
b) If the independent variables are highly correlated, the accuracy of the model may be negatively impacted.
c) Linear regression models can be vulnerable to noise and overfitting, which may result in inaccurate predictions.

Implementation:

Training vs Testing Score for Linear regression for min_salary

	Accuracy_training	R2 Score_training	RMSE_training	Accuracy_test	R2 Score_test	RMSE_test
0	0.341448	0.341448	27878.544454	0.282702	0.282702	27924.942326

Testing Score for Linear regression for max_salary

	Accuracy_training	R2 Score_training	RMSE_training	Accuracy_test	R2 Score	RMSE
0	0.275463	0.275463	38016.807783	0.225626	0.225626	39786.189849

2. Decision tree regression

Decision tree regression is an ML algorithm that uses a tree-like structure to model a problem. The algorithm works by recursively dividing the dataset into smaller subsets based on the value of a particular attribute. Each

split is designed to maximize the information gain, which is the difference between the entropy of the parent node and the sum of entropies of the child nodes. The goal of the algorithm is to create a model that can accurately predict a continuous output variable (in our case, average salary). At each node, the algorithm selects the attribute that provides the most information gain, meaning the attribute that best splits the data into subsets that are as different as possible from each other in terms of the output variable. The process continues until a stopping criterion is met, such as when a minimum number of instances are reached, or the maximum depth of the tree is achieved. At this point, the leaf nodes represent the predicted value for the output variable.

Advantages:

a) This is another algorithm which is easy to understand as well as interpret, even for a layman.
b) It can handle both numerical as well as categorical data.
c) It can capture non-linear relationships between the input variables and the target variable.
d) Decision trees can also handle missing values and outliers in an effective manner.

Disadvantages:

a) Decision trees are also prone to overfitting.
b) Sometimes, making small changes in the data can cause large changes in the tree structure, therefore, decision trees are quite unstable.
c) They could also be biased towards independent variables that have more number of levels or values, so decision trees are not always reliable.
d) They can be sensitive to the order of the input variables, which can affect the tree structure, and ultimately affect the prediction performance.

Implementation:

Training VS Testing Score for decision tree regressor for minimum salary

	Accuracy train	R2 Score train	RMSE train	Accuracy_test	R2 Score test	RMSE test
0	0.483564	0.483564	24687.860538	0.431993	0.431993	24849.592498

Training VS Testing Score for decision tree regressor for maximum salary

	Accuracy train	R2 Score train	RMSE train	Accuracy_test	R2 Score test	RMSE test
0	0.434335	0.434335	33591.170519	0.440809	0.440809	33809.376441

3. Random Forest regression

Random Forest regression is an ML algorithm that can also be used for the prediction of salaries. It combines ensemble learning with decision trees to create many random decision trees (which are used as base learning models) drawn from the dataset, averaging the results to give a new output that leads to stronger predictions. It uses a technique called “bootstrap and aggregation”, otherwise known as “bagging”.

Advantages:

a) This model can also handle both numerical and categorical data.
b) It captures the non-linearity between the input and output variables, and it also handles missing values and outliers effectively.

c) In comparison with decision tree regression, it reduces the risk of overfitting by using an ensemble of multiple decision trees and a random sampling of input variables and training observations.

Disadvantages:

a) RF regression is computationally expensive as well as memory expensive, especially for large datasets like the one we're using.
b) It is more difficult to understand in comparison with decision tree regression because of the use of multiple trees and feature important measures.
c) It can be biased towards certain variable(s), especially when there are strong correlations between them.
d) The curse of dimensionality also comes into play in case of RF regression, that is, the performance of the model might degrade as the number of input variables increases.

Implementation:

Training vs Testing Score for Random Forest regressor for minimum salary

	Accuracy train	R2 Score train	RMSE train	Accuracy_test	R2 Score test	RMSE test
0	0.991921	0.991921	3087.907958	0.927122	0.927122	8901.02863

Training vs Testing Score for Random Forest regressor for maximum salary

	Accuracy train	R2 Score train	RMSE train	Accuracy_test	R2 Score test	RMSE test
0	0.978316	0.978316	6576.864288	0.855392	0.855392	17193.043761

4. AdaBoost regression

One of the first ever boosting algorithms, AdaBoost helps in combining multiple weak learners into a single strong learner. Adaptive boosting is used as an ensemble method. The most common estimator used with AdaBoost is decision trees with one level which means Decision trees with only 1 split. These trees are also called Decision Stumps. What this algorithm does is that it builds a model and gives equal weights to all the data points. It then assigns higher weights to points that are wrongly classified. Now all the points with higher weights are given more importance in the next model. It will keep training models until and unless a lower error is received.

Advantages:

a) AdaBoost works well with a large number of predictors and automatically detects interactions between them. This feature makes it particularly useful for predicting salaries, where numerous factors like education, experience, job type, and location may interact.
b) Despite being a complex algorithm, AdaBoost is fast and efficient, making it suitable for large datasets.
c) AdaBoost can reduce overfitting compared to decision tree regression by using an ensemble of multiple weak models and adaptive weighting of the training instances.
d) AdaBoost is a flexible algorithm that can be adapted to different types of data and classification problems.

Disadvantages:

a) AdaBoost is highly sensitive to noisy data and outliers, which can negatively affect the performance of the weak models and the ensemble as a whole. It is, therefore, crucial to clean and preprocess data before using AdaBoost.
b) AdaBoost can be computationally expensive for large datasets as it involves iterative training of multiple models.
c) AdaBoost can be difficult to interpret, making it challenging to explain its results to non-experts.
d) AdaBoost may be biased towards certain variables/predictors when strong interactions exist between them, which can result in unreliable predictions. This issue can be mitigated by using techniques like regularization or feature selection.

Implementation:

Training vs Testing Score for Adaboost regressor for min salary

	Accuracy train	R2 Score train	RMSE train	Accuracy_test	R2 Score test	RMSE test
0	0.623991	0.623991	21065.616116	0.568453	0.568453	21659.908526

Training vs Testing Score for Adaboost regressor for max salary

	Accuracy train	R2 Score train	RMSE train	Accuracy_test	R2 Score test	RMSE test
0	0.509741	0.509741	31272.190385	0.48286	0.48286	32513.310565

5. Support Vector Regression

Support vector regression (SVR) is an ML algorithm that can be used for predicting the average salary. It is based on the Support Vector Machine (SVM) algorithm and it is a type of regression analysis where the goal is to predict the value of a continuous target variable (here, average salary) based on multiple input variables. In SVR, the algorithm tries to find the best-fitting line or hyperplane that can predict the response variable while minimising the error. It uses a kernel function to transform the input data into a higher dimensional space where a linear model can be used to predict the target variable.

Advantages:

a) It is a very powerful algorithm that can handle both non-linear data as well as high dimensional data. It is hence very useful for predicting salaries as the relationship between the average salary and the predictors can be quite complex or non-linear.
b) It is a robust algorithm that can handle outliers in the data as well. Outliers in salary data might arise due to factors like bonuses and incentives, and SVR can handle them pretty effectively.
c) SVR has good generalization performance, therefore, it can perform well on new data.
d) The kernel function used in SVR can be customized to different types of data, making it flexible for different applications.

Disadvantages:

a) It can be computationally expensive for a large dataset like ours, with high dimensional data. It requires more computing power and time compared to other regression algorithms.
b) The choice of kernel function can have a significant impact on the performance of the algorithm, and it may not always be clear which kernel function to use for a given dataset.

c) It can be sensitive to the choice of hyperparameters, such as the regularization parameter and the kernel function parameters. This sensitivity can make it challenging to find the optimal hyperparameters for a given dataset.
d) Interpretation of the results is difficult, as it may be difficult to explain the relationship between the input variables and the predicted salary.

Implementation:

```
Testing Score for Support vector regressor
   Accuracy  R2 Score         RMSE
0  0.017193  0.017193  35567.42367
```

Inference from the above models

Although we used 149 predictor variables for our models, their accuracy scores were observed to be low for most models, except the random forest regression model, which produced an accuracy of 92.7% and 85.5% for minimum and maximum salary respectively. This is likely due to the high variance in our data caused by a large number of variables. However, we plan to improve our data processing and manipulation techniques to reduce the number of predictor variables and increase the accuracy of the models. We will also perform hyperparameter tuning to optimize the performance of the models, ultimately leading us to select the best model for our project.

Analyzing Data: Results, Inferences, and Interpretation in the above prediction models

The result from regression models:
Based on the evaluation of different prediction models, it appears that the **random forest regression** model has shown better **accuracy (92.7%, 85.5%)** as well as R2 score on the testing data compared to others. However, most prediction models have shown very low accuracy values. **Hence, we have decided to switch to classification for our dataset.**

Interpretation:
We initially tried to solve the problem using regression models, but due to the limitations of the dataset (way too many variables and a **very less number of unique values for minimum and maximum salaries, when compared to the total size of the dataset**), we were getting low R2 scores and accuracies. We, therefore, decided to reframe the problem as a classification one and **split the mean salaries into three categories: low, average, and high**, based on the first and third quartiles. We used this new target variable to train several different classification algorithms, including Decision Trees, Random Forests, Support Vector Machines, K-Nearest Neighbours, Naive Bayes, and Gradient Boosting.

These classification algorithms allowed us to predict the salary range for a given job based on its title, description, rating, and other company information. By doing so, we were able to gain insights into the general salary trends in our dataset and make more informed hiring and compensation decisions. Additionally, the simplicity of the classification models made them more interpretable and easier to communicate to stakeholders.

Our regression models were not effective, as they were not able to account for the complexity of the dataset and were not able to learn enough from the data. However, by reframing the problem as a classification one, we were able to create a simpler, more effective model that was better suited to our dataset and our needs. The combination of different classification algorithms allowed us to explore different ways of solving the problem and choose the most effective one. Overall, our approach allowed us to overcome the limitations of the original dataset and create a more useful and interpretable model that can be used to inform decision-making in real-world applications.

CLASSIFICATION MODELING

Performance Evaluation on the Classification Models

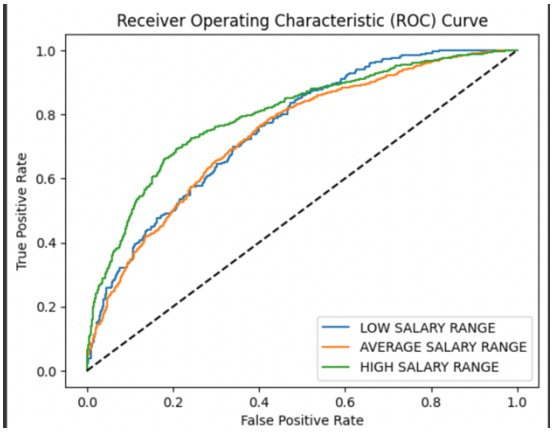
```
Class 0 Corresponds to LOW SALARY RANGE
Class 1 Corresponds to AVERAGE SALARY RANGE
Class 2 Corresponds to HIGH SALARY RANGE
```

1. Logistic Regression

Logistic Regression is a linear classification algorithm that uses a sigmoid function to predict the probability of an event occurring. It works well when the decision boundary is linear and the features are not highly correlated. It is widely used for binary classification problems and can be easily interpreted.

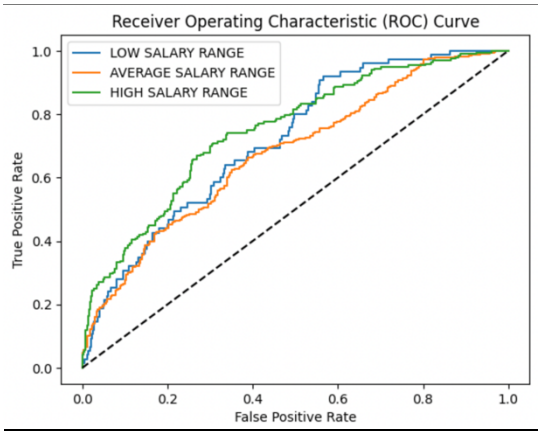
Training:

```
****** For Training on  Logistic Regression Model ******

The Confusion Matrix:
 [[ 12 213  31]
 [  3 979 182]
 [  0 269 422]]

 The Accuray Score: 0.6693510184746566

 The Recall Score: 0.6107091172214182

 The Precision Score: 0.6645669291338583

 The F1 Score: 0.636500754147813

 The ROC AUC Score: 0.6396760743110989
```



Testing:

```
****** For Testing on Logistic Regression Model ******

The Confusion Matrix:
 [[  2  66   7]
 [  6 286  68]
 [  0 103  93]]

 The Accuray Score: 0.6038034865293186

 The Recall Score: 0.4744897959183674

 The Precision Score: 0.5535714285714286

 The F1 Score: 0.5109890109890111

 The ROC AUC Score: 0.5814632518095323
```



2. Decision Tree Classifier

Decision Tree Classifier is a non-linear algorithm that uses a tree-like model of decisions and their possible consequences. It works by recursively splitting the data into smaller subsets based on the most significant features until a homogeneous subset is achieved. It can handle both numerical and categorical data and is often used in problems where the decision boundary is non-linear and complex.

Training:

```
****** For Training on  Decision Tree Classifier Model ******

The Confusion Matrix:
 [[ 256    0    0]
 [   0 1164    0]
 [   0    0  691]]

 The Accuray Score: 1.0

 The Recall Score: 1.0

 The Precision Score: 1.0

 The F1 Score: 1.0

 The ROC AUC Score: 1.0
```

Testing:

```
****** For Testing on Decision Tree Classifier Model ******

The Confusion Matrix:
 [[ 69   3   3]
 [  4 343  13]
 [  0   7 189]]

 The Accuray Score: 0.9524564183835182

 The Recall Score: 0.9642857142857143

 The Precision Score: 0.9219512195121952

 The F1 Score: 0.942643391521197

 The ROC AUC Score: 0.9593645448766805
```

3. Random Forest Classifier

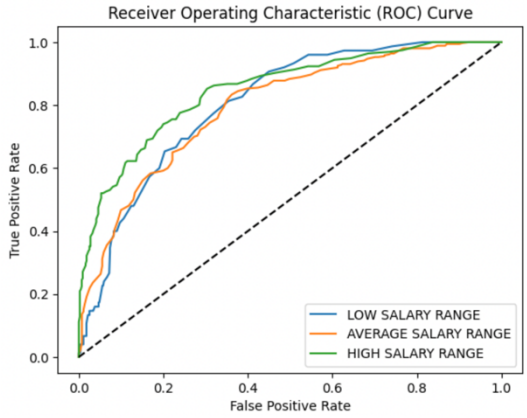
Random Forest Classifier is an ensemble learning algorithm that combines multiple decision trees to improve the accuracy and reduce overfitting. It works by randomly selecting a subset of features and samples to create a large number of decision trees, and then aggregating the predictions of these trees to make the final prediction. It is widely used for both classification and regression problems and is known for its robustness and high accuracy.

Training:

```
****** For Training on  Random Forest Classifier Model ******

The Confusion Matrix:
 [[ 256    0    0]
 [   0 1164    0]
 [   0    0  691]]

 The Accuray Score: 1.0

 The Recall Score: 1.0

 The Precision Score: 1.0

 The F1 Score: 1.0

 The ROC AUC Score: 1.0
```

Testing:

```
****** For Testing on Random Forest Classifier Model ******

The Confusion Matrix:
 [[ 13  57   5]
 [ 18 310  32]
 [  7  76 113]]

 The Accuray Score: 0.6909667194928685

 The Recall Score: 0.576530612244898

 The Precision Score: 0.7533333333333333

 The F1 Score: 0.6531791907514451

 The ROC AUC Score: 0.6650297748164865
```

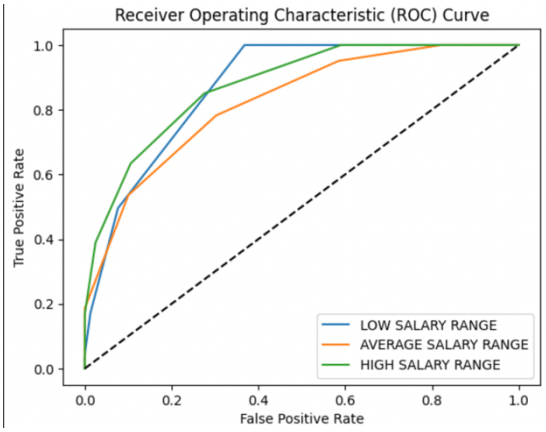


4. K-Neighbours Classifier

K-Nearest Neighbor Classifier is a simple, non-parametric algorithm that classifies a new instance based on the majority class of its k-nearest neighbors in the training set. It works well when the decision boundary is non-linear and the data is not highly dimensional. It is often used for problems where the data is spatially correlated, such as image classification and natural language processing.

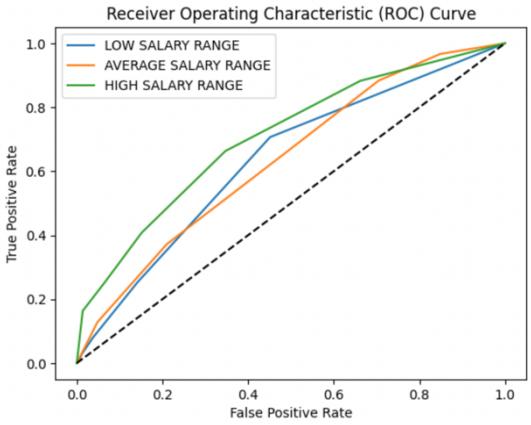
Training:

```
****** For Training on  KNN Classifier Model ******

The Confusion Matrix:
 [[ 92 145  19]
 [ 67 966 131]
 [ 30 223 438]]

 The Accuray Score: 0.708668877309332

 The Recall Score: 0.6338639652677279

 The Precision Score: 0.7448979591836735

 The F1 Score: 0.684910086004691

 The ROC AUC Score: 0.712769233267351
```



Testing:

```
****** For Testing on KNN Classifier Model ******

The Confusion Matrix:
 [[ 14  55   6]
 [ 41 259  60]
 [ 15 101  80]]

 The Accuray Score: 0.5594294770206022

 The Recall Score: 0.40816326530612246

 The Precision Score: 0.547945205479452

 The F1 Score: 0.46783625730994155

 The ROC AUC Score: 0.5810308429280339
```

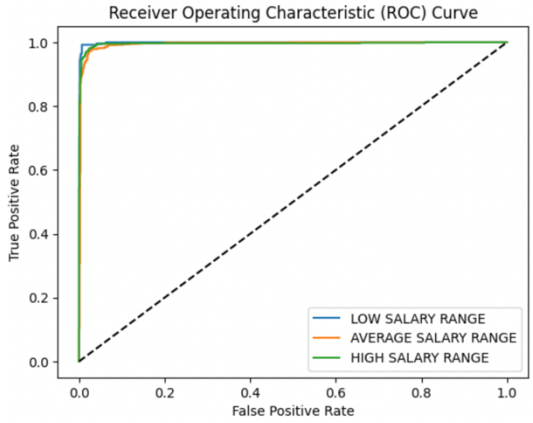


5. Gradient Boosting Classifier

Gradient Boosting Classifier is another ensemble learning algorithm that combines multiple weak predictors to form a strong predictor. It works by iteratively adding decision trees to the model, with each tree correcting the errors of the previous tree. It is known for its high accuracy, especially in problems where the features are highly correlated and the decision boundary is complex.

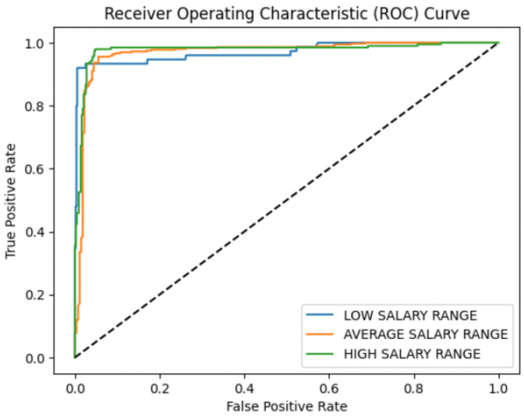
Training:

```
****** For Training on  Gradient Boosting Classifier Model ******

The Confusion Matrix:
 [[ 224   32    0]
 [   2 1154    8]
 [   1   46  644]]

 The Accuray Score: 0.9578398863098058

 The Recall Score: 0.9319826338639653

 The Precision Score: 0.9877300613496932

 The F1 Score: 0.9590469099032018

 The ROC AUC Score: 0.9514625251287953
```



Testing:

```
****** For Testing on Gradient Boosting Classifier Model ******

The Confusion Matrix:
 [[ 58  14   3]
 [  2 349   9]
 [  0  20 176]]

 The Accuray Score: 0.9239302694136292

 The Recall Score: 0.8979591836734694

 The Precision Score: 0.9361702127659575

 The F1 Score: 0.9166666666666666

 The ROC AUC Score: 0.914015396273332
```



Project Results

1. The Decision Tree Classifier has the highest overall accuracy of 95.2%. It also has the highest F-1 score of 94.2%. It has the highest ROC value of 95.93%, which implies that the Decision Tree Classifier is the best model among all the other models in classifying all three classes of salaries accurately.

2. The Gradient Boosting Classifier is the next best classifier in terms of accuracy with a value of 92.3%. It has a high F1 score value of 91.7%.

3. The Random Forest Classifier is the next best classifier in terms of accuracy with a value of 69.1%. It has a high F1 score value of 65.3%.

4. The Logistic Regression Classifier is the next best classifier in terms of accuracy with a value of 60.3%. It has an F1 score value of 51.1%.

5. The K-Neighbours Classifier is the least-performing classifier in terms of accuracy with a value of 55.9%. It has the lowest F1 score value of 46.8%.

Evaluating Overfitting and Underfitting

It is evident that there is no underfitting or overfitting as training and testing errors are not far apart for most of the models.

Classification Model	Training Error	Testing Error
K-Neighbours Classifier	29.2%	44.1%
Logistic Regression Classifier	33.1%	39.7%
Random Forest Classifier	0%	30.9%
Gradient Boosting Classifier	4.3%	7.7%
Decision Tree Classifier	0%	4.8%

Conclusion

Decision Tree Classifier is a powerful machine learning algorithm that is known for its ability to achieve high accuracy in classification tasks. In the context of our project, the Decision Tree Classifier outperformed other models because it was able to effectively combine multiple weak predictors to form a strong predictor. Additionally, Decision Tree Classifier is robust to overfitting and can handle many features, making it an ideal choice for our dataset, which consisted of job titles, job descriptions, company size, company age, job location, company competitors, and other demographic information. The high accuracy achieved by our model can be attributed to the fact that this algorithm considers the interactions between different features, allowing it to capture complex relationships in the data.

Impact of the Project Results

The impact of our project on society can be significant, as it can provide valuable insights to job seekers and employers alike. By accurately predicting salary ranges for different data jobs, our model can help job seekers

make informed decisions about their career paths, negotiate salaries, and improve their overall financial stability. Additionally, employers can use this information to attract and retain top talent, as well as ensure that their compensation packages are competitive in the market. Overall, our project can contribute to a more efficient and transparent job market, which can benefit both individuals and society.

In addition to the direct impact on job seekers and employers, our project can also have broader societal implications. By shedding light on the factors that influence salaries in the data job market, we can uncover potential biases and disparities that may exist. For instance, our model may reveal that certain demographic groups are systematically paid less than others, even after controlling for relevant factors such as education and experience. This information can help identify areas for policy intervention and advocacy efforts aimed at promoting equity and fairness in the labour market.

Furthermore, our project can serve as a template for similar analyses in other industries and domains. The approach we used, which involved combining multiple sources of data and applying machine learning techniques, can be adapted to a wide range of applications, from healthcare to finance to education. By demonstrating the potential of data-driven methods to inform decision-making and improve outcomes, we can inspire others to explore similar avenues and contribute to the advancement of the field. Ultimately, our project underscores the power of data and analytics to drive social and economic progress, and we hope it inspires others to join us in this pursuit.