"Dissemination of Education for Knowledge, Science and Culture."

-Shikhanmaharshi Dr. Bapuji Salunkhe

VIVEKANAND COLLEGE, KOLHAPUR

(Empowered Autonomous)

DEPARTMENT OF STATISTICS

A PROJECT REPORT

ON

"Skill-Based Job Recommendation System Using Machine Learning Techniques."

Submitted by

Mr. Chavan Shubham Suresh
Mr. Parit Amol Dhanaji
Ms. Alavane Rutuja Maruti

in partial fulfillment for

the award of the degree of

MASTER OF SCIENCE

in

STATISTICS

2023-24

" Dissemination of Education for Knowledge, Science and Culture."

-Shikhanmaharshi Dr. Bapuji Salunkhe

VIVEKANAND COLLEGE, KOLHAPUR

(Empowered Autonomous)

DEAPRTMENT OF STATISTICS

Certificate

This is to Certify that,

Sr. No.	Name	Roll No.
1	Mr. Chavan Shubham Suresh	1402
2	Mr. Parit Amol Dhanaji	1414
3	Ms. Alavane Rutuja Maruti	1401

Have satisfactorily completed the project work on **"*Skill-Based Job Recommendation System using machine learning techniques.*"** as a part of practical evaluation course for ***M.Sc. II,*** prescribed by the Department of Statistics, ***Vivekanand College, Kolhapur (Empowered Autonomous)*** in the academic year **2023-24**.

This project has been completed under our guidance and supervision. To the best of our knowledge and belief, the matter presented in this project report is original and has not been submitted elsewhere for any other purpose.

Project Guide	Examiner	Head
(Ms. Pandhare R. S.)		(Mrs. Shinde V. C.)

ACKNOWLEDGEMENT

I am privileged to express my sincere thanks with great respect and gratitude to Mrs. V. C. Shinde (Head, Department of Statistics), Miss. Pandhare R. S. and all staff members for their aspiring guidance. They all helped with kind of co-operation and constant encouragement. I am grateful to thank them for providing me with all necessary facilities.

Also, I would like to thank all the non-teaching staff of the department for their help and co-operation. I thank all my friends and the teaching staff for their co-operation and help which I received from them during the work throughout. I am indebted to my parents for their encouragement and patience throughout my study and for the trust they have on me.

Yours Sincerely,

M Sc II

Department of Statistics

INDEX

INTRODUCTION

We know that, now a days getting a job is being a difficult and challenging. Most of the people don't know how to find a job and which job is suitable for us according to their skills. A recent report says that most college, graduates and Post graduates have difficulty in choosing their domain in their job. Now a days jobs in the IT field are increases hence many students are trying to shift the domain from their field to IT. So, they are doing some courses online and randomly searching for a job.

Nowadays, IT fields are the most preferable job of many students but they don't know which job is fit for them. To avoid this situation candidates, need a Job recommendation that analyses the skills to recommend a suitable job for the candidate. The solution is to design a system that reads a resume and their skills. The resumes are going through pre-processing to make the design more efficient. For pre-processing top words and porter stemmer, Porter Stemmer will make every word their root word, and stop words will remove every meaningless word. This makes the system more efficient. Then compare the skills in the resume and description.

The key problem is that most job-hunting websites just display recruitment information to website viewers. Students have to go through all the information to find the jobs they want to apply for. The whole procedure is tedious and inefficient. We need an easy job recommendation system where everyone will have a fair and square chance. This saves a lot of potential time and money both, on the industrial as well as the job seeker's side. Moreover, as the candidate gets a fair chance to prove his talent in the real world it is a lot more efficient system. The basic agenda of every algorithm used in today's world, be it a traditional algorithm or a hybrid algorithm, is to provide a suitable job that the user seeks and wishes for.

The main purpose this project is to work focuses on predicting suitable jobs for the candidates as per acquire skills. It uses machine learning models to find similarities between jobs description and resumes to predict accurately. This application can be used by any candidates who need or who want to know about their suitable jobs and to improve themselves with both soft skills and hard skills. It will be helpful to them by not waste their time searching for jobs and prefer a job as per acquire skills.

MOTIVATION

Nowadays we are at a point where we are standing with huge data collected with us. As a statistician, our job is to study and analyze the data. Currently, many students pass out their degree or post-degree course and this job seeker wants a job but they don't know exactly what type of skills are needed for a particular job and if so what kind of skills they have. In this project, we have considered three main Job types Data Analyst, Data Engineer, and Data Scientist. We are also looking at whether the job type and salary depend on the location.

The main purpose of our study was to handle big data. To study an association between Job type and skills. The number of skills or the mainly specific skills required for particular jobs.

Many job seekers are trying to shift the domain from their field to IT. So, they are doing some courses online and randomly searching for a job. Nowadays, IT fields are the target of many students but they don't know which domain is suitable for them. To avoid this situation candidate need a Job Recommendation that analyses the skills to recommend a suitable job for the candidate. The solution is to design a system that reads a resume and their skills. The resumes are going through pre-processing to make the design more efficient.

The proposed work focuses on predicting suitable jobs for the candidates. It uses machine learning models to find similarities between jobs description and resumes to predict accurately. This application can be used by any candidates, who need or who want to know about their suitable jobs and to improve themselves with both soft skills as well as hard skills. It will be helpful to them by not waste their time searching for jobs. They can also grow their skills in their domain and grow faster in their domain.

OBJECTIVE

- To build a recommendation system that will help candidates to get suitable jobs concerning their skills.
- To suggest skills to the candidates so that they can acquire them and get a suitable job.
- To check which factors are significant and which factors affect our target variable.
- To check which type of Job candidates can prefer the most among the job type viz. Data Scientist, Data Analyst, and Data Engineer.

METHODOLOGY

We have collected our data from Kaggle the data contain. Data Scientist, Data Analyst, and Data Engineer. To find suitable jobs and their scores, this application receives the resume and has a dataset for a job with their description. Data mining is a field of the intersection of computer science and statistics used to discover patterns in the information bank. The main goal of the data mining process is to extract useful information from the dossier of data and mold it into an understandable structure for future use. The system will move on to the next progress which is finding the skills to be improved by the candidates. The system will take the resume and the skills dataset then compares both and display the skills which are all not in the resume. The methods used are Random Forest, Decision Tree, KNN, and Naive Bayes, for the ensemble so that even if one method predicts incorrectly, the other model is likely to make the correct prediction, and since the majority voting technique used for the final prediction is the correct one. In this project we use a **Multinomial logistic regression** is an extension of logistic regression that adds native support for multi-class classification problems.

These data will be in an unstructured format, so we need to pre-process the data before going to the modeling stages. Once the extracted data is processed and transformed in to feature, we can further go ahead and utilize those features as input to the content-based filtering. Then we feed the user and item feature matrix to a similarity measure algorithm to identify the similarity between the job and the user.

Now we take an X as an independent variables and Y is dependent variable as Job Type. Let us consider the sklearn model selection by splitting the data as train and test datasets. In this model we take a test size is 0.2 i.e. the train data is 80% and the test data set is 20%. Then we apply the classification algorithm and build the models and check which model is best fit for our data.

Data description

- We can easily tell that there is a mix of categorical, geographical, and numerical variables.
- Each row represents the candidate's information like no. of skills, Salary, company, revenue and employee size, location, and the field they work in.
- There are 5715 rows including the header and 43 columns. The data type is correct and matchesthe corresponding values.

Data Description: -

- **Job_Title**: Title of Role
- **Link**: Weblink of Job Posting
- **Queried_Salary**: Salary Range of the Job Posting (Estimated/Actual if available)
- **Job_Type**: **3 Categories of Job Types - Data_Scientist, Data_Analyst, Data_Engineer**
- **Skill**: List of desired skills on the Indeed site
- **No of Skill**: Count the number of desired skills
- **Company**: The company that posted the job posting
- **No of Reviews**: Number of Reviews for the Company
- **No of Stars:** Ratings for the Company
- **Date Since Posted**: Number of days since the job was posted - in less than a day
- **Description**: Web scrape of part of the job description
- **Location**: State the job opening is located in
- **Company_Revenue**: Annual revenue of hiring company
- **Company_Employees**: Employee count of hiring company
- **Company_Industry:** Industry of hiring company

Expansion of Skill column (Top 10 overall skills):

- python
- sql
- machine learning
- r
- hadoop
- tableau
- sas
- spark
- java
- Others

Expansion of Location column (Top 10 overall states):

- CA
- NY
- VA
- TX
- MA
- IL
- WA
- MD
- DC
- NC
- Other_states

Expansion of Company_Industry columns (Top 5 overall industries)

- Consulting and Business Services
- Internet and Software
- Banks and Financial Services
- Health Care
- Insurance, Other_Industries.

Data Pre-processing

Data Cleaning:

Before we start with the analysis, we must first clean the data or “scrub the dirt”. For this analysis, we will look at the more common issues such as missing and duplicate data.

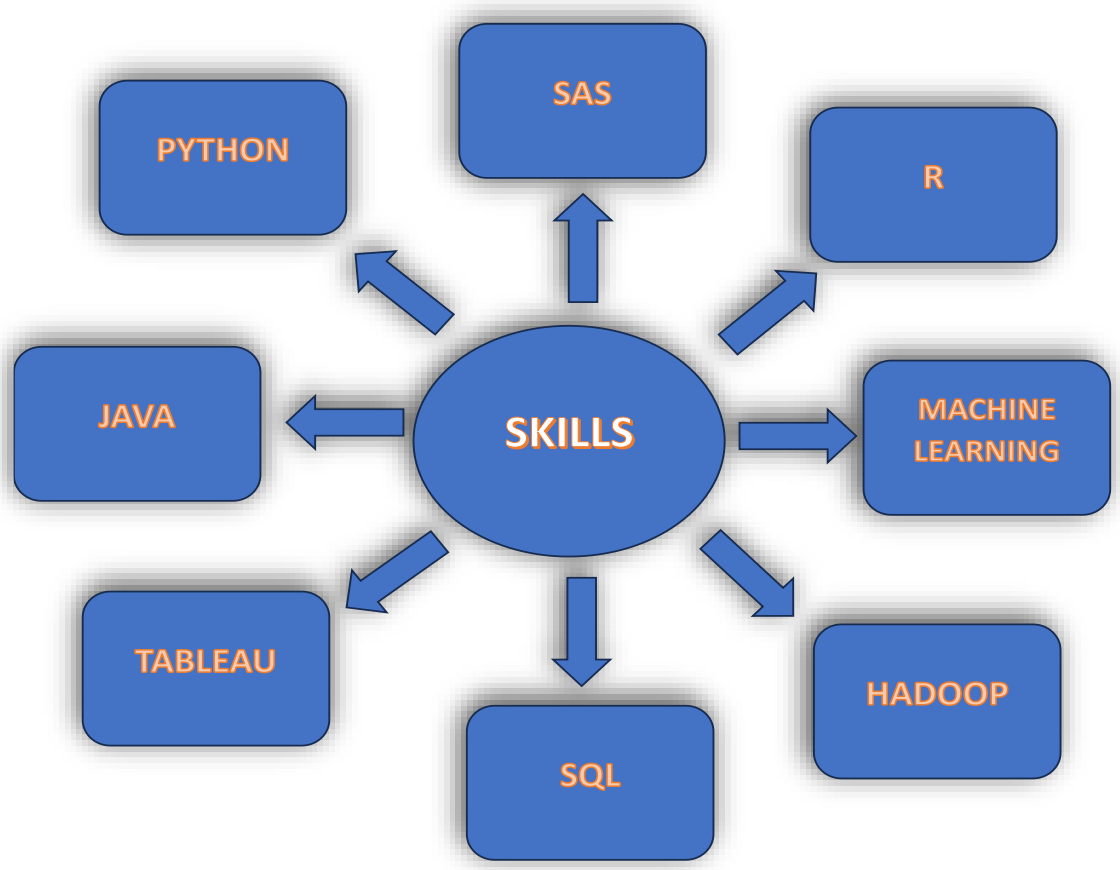
Handling Missing Values: There are no missing values.

Duplicate Data:

Then, we will find out whether there is duplicate data.The result shows that there are no duplicated rows.

Now that the data set has been scrubbed, we can proceed with some statistics analysis!

SKILLS:



Exploratory Data Analysis

Bar Plot for which job prefers Most:

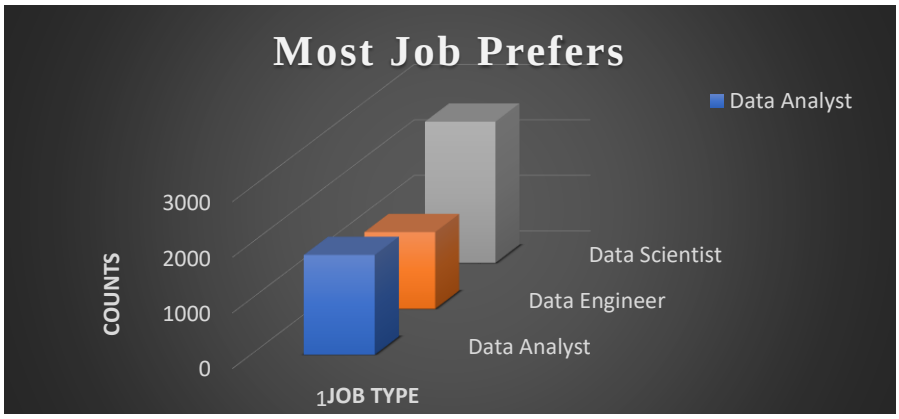


Fig. Which job prefers most of the peoples

Conclusion:

From above figure, we can see that Data Scientist job prefers by 45% peoples. Data analyst prefers 31% and Data Engineer prefers 24% of peoples.

Histogram:

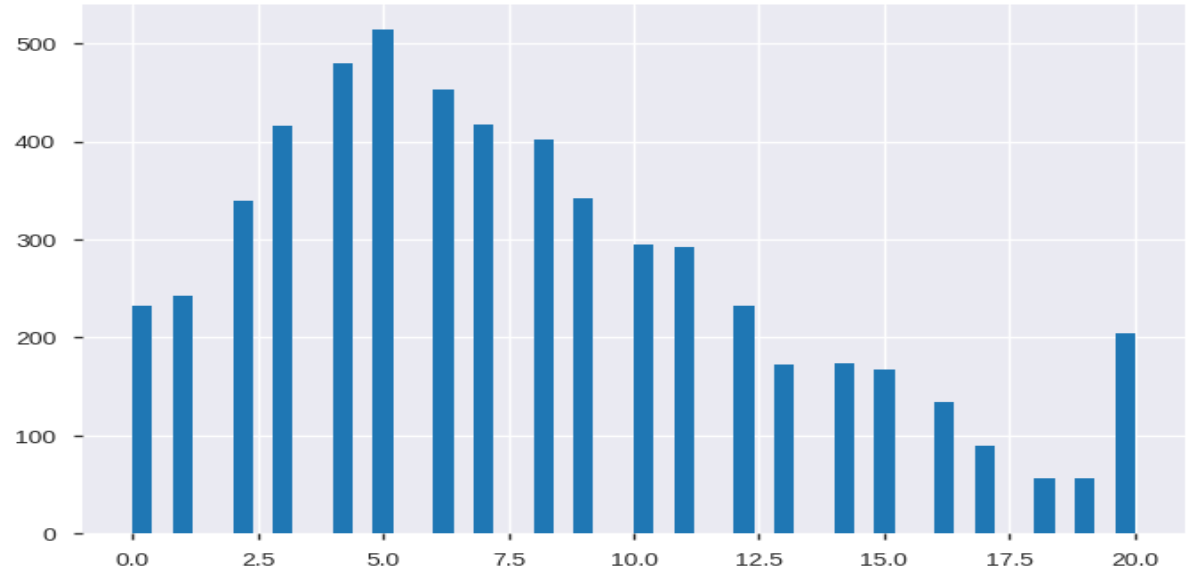
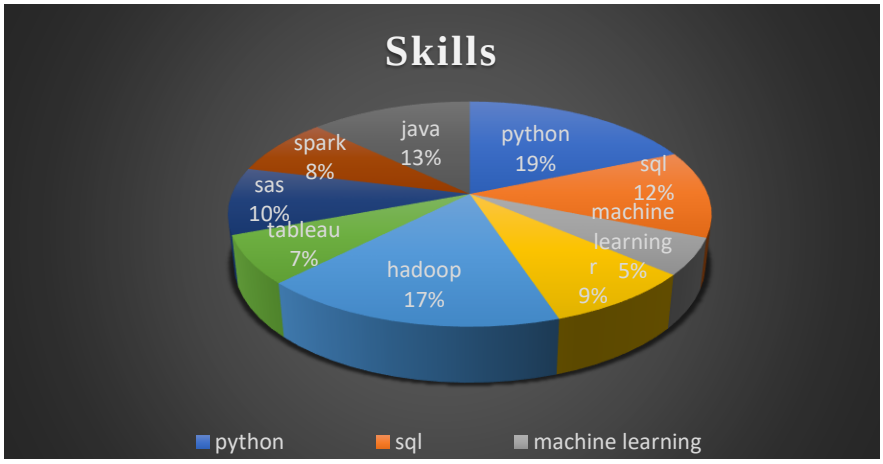


Fig. Distribution of the Numbers of Skills acquire

Conclusion:

- We can observe that No of Skills is positively skewed.
- The most no. of skills candidates have is 5.

Pie Chart:



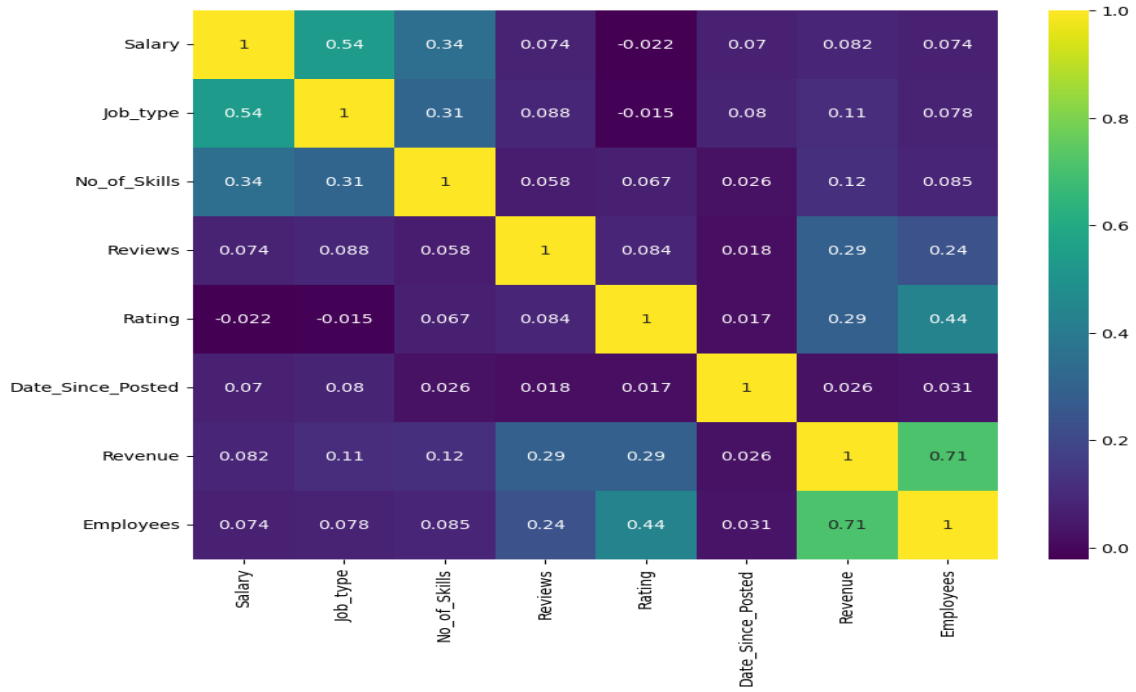
Conclusion:

- Most of the **19%** of candidates are likely to have **Python** skills.
- Second **12%** of candidates are likely to have **SQL** skills.
- And then **5%** of candidates are likely to have **machine learning** skills.

Correlation Matrix:

	Salary	**Job type**	**No. of skills**	**Reviews**	**Rating**	**Date since posted**	**Revenue**	**Employees**
Salary	1.00000	0.535437	0.344955	0.074453	0.021550	0.070500	0.082360	0.073881
Job type	0.535437	1.000000	0.307063	0.087881	0.015435	0.080042	0.110028	0.077899
No. of skills	0.344955	0.307063	1.000000	0.058131	0.066961	0.025671	0.121564	0.085359
Reviews	0.074453	0.087881	0.058131	1.000000	0.083923	0.018019	0.287024	0.235445
Rating	0.021550	0.015435	0.06961	0.083923	1.000000	0.017314	0.287510	0.436637
Date Since posted	0.070500	0.080042	0.025671	0.018019	0.017314	1.000000	0..026048	0.031391
Revenue	0.082360	0.110028	0.121564	0.287024	0.287510	0.026048	1.000000	0.712080
Employees	0.073881	0.077899	0.085359	0.235445	043663 7	0.031391	0.712080	1.0000000

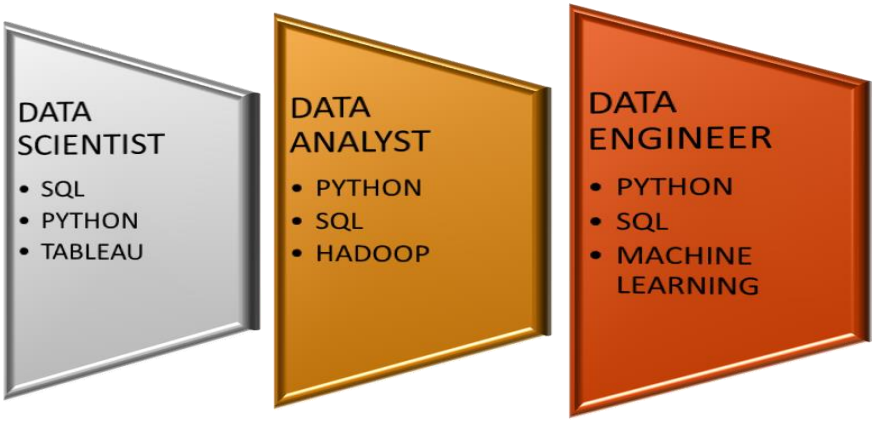
Heatmap:



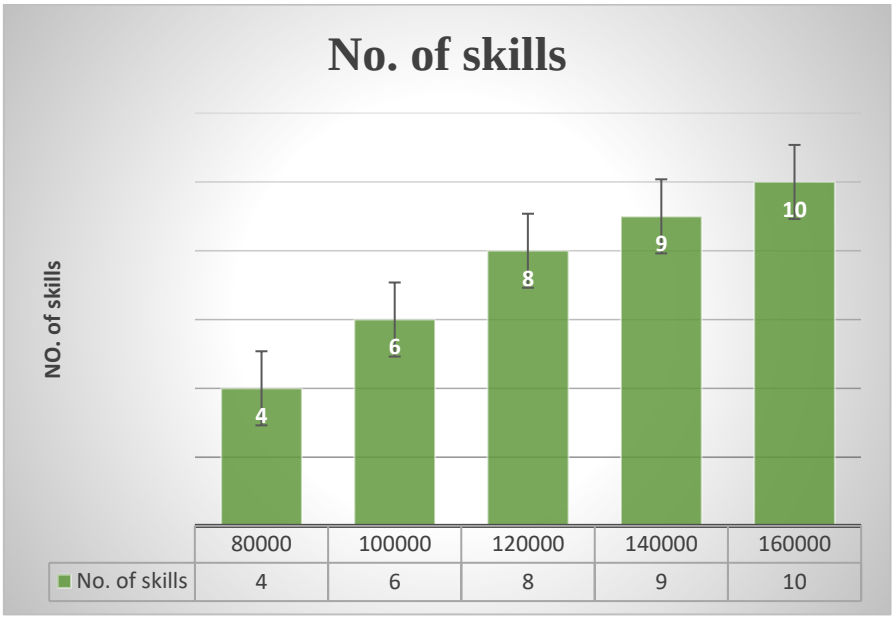
Conclusion:

- From the correlation matrix and heatmap, we observe that most of the variables are less correlated with each other.
- Thus, the multicollinearity is absent in the data.
- We can see that employee and revenue have the highest correlation. i.e. employee and revenue mostly depend on each other.
- There is no relation between company rating and the salary of the candidate.

Top Skills are required for particular job:



Relation between Number of skills and Salary:



Conclusion:

- Salary is dependent on Number of skills.
- If we acquire minimum of 10 skills, then we can get a salary up to Rs.160000 per month.
- If we acquire minimum of 4 skills, then we can get a salary up to Rs.80000 per month.

Statistical Test

Normality Test:

In statistics, normality tests are used to determine if a data set is well-modeled by a normal distribution and to compute how likely it is for a random variable underlying the data set to be normally distributed.

An important decision point when working with a sample of data is whether to use parametric or nonparametric statistical methods.

Parametric statistical methods assume that the data has a known and specific distribution, often a Gaussian distribution. If a data sample is not Gaussian, then the assumptions of parametric statistical tests are violated and nonparametric statistical methods must be used.

There are a range of techniques that you can use to check if your data sample deviates from a Gaussian distribution, called normality tests.

In this tutorial, you will discover the importance of checking whether a data sample deviates from the normal distribution and a suite of techniques that you can use to evaluate your data sample

Conclusion:

Before you can apply the statistical tests, you must know how to interpret the results.

Each test will return at least two things:

- **Statistic**: A quantity calculated by the test that can be interpreted in the context of the test by comparing it to critical values from the distribution of the test statistic.
- **p-value**: Used to interpret the test, in this case, whether the sample was drawn from a Gaussian distribution.

Each test calculates a test-specific statistic. This statistic can aid in the interpretation of the result, although it may require a deeper proficiency with statistics and a deeper knowledge of the specific statistical test. Instead, the p-value can be used to quickly and accurately interpret the statistic in practical applications.

The tests assume that the sample was drawn from a Gaussian distribution. Technically this is called the null hypothesis, or H0. A threshold level is chosen called alpha, typically 5% (or 0.05), which is used to interpret the p-value.

Test Statistics:

p ≤ alpha: reject H0, not normal.

p > alpha: fail to reject H0, normal.

Statistical Hypothesis:

H_0: The data is normally distributed.

V/s

H_1: The data is not normally distributed.

A result above 5% does not mean that the null hypothesis is true. It means that it is very likely true given the available evidence. The p-value is not the probability of the data fitting a Gaussian distribution; it can be thought of as a value that helps us interpret the statistical test.

Conclusion:

From the Normality test, we conclude that the dataset is normally distributed.

Chi-square Test:

Chi- square Test Analysis:

Testing independence of salary and numbers of skills acquire

Hypothesis:

H0: Salary is independent on number of skills acquire.

H1: Salary is not independent on number of skills acquire.

Chi-square statistic: 468.4970, α=0.05, p-value: 0.4014e-94

p-value < 0.05

Therefore, Reject H0 at 5% level of significance.

Conclusion:

Salary is dependent on number of skills acquire.

CLASSIFICATION ALGORITHMS

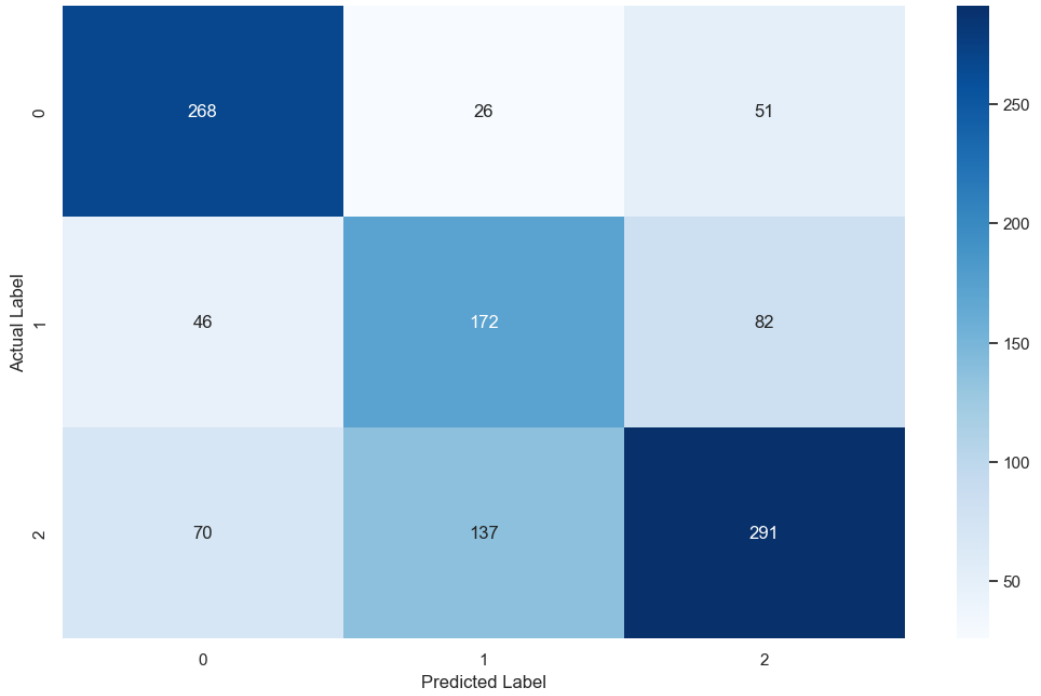
K -Nearest Neighbor Classifier (KNN)

K nearest neighbor, also known as the KNN algorithm, is a non-parametric algorithm that classifies data points based on their proximity and association with other available data. This algorithm assumes that similar data points can be found near each other and classify the new data on the basic of similarity.

Classification Report:

Precision	Recall	F1-Score	Accuracy	Roc Curve
0.63	0.63	0.64	0.63	0.84

Confusion Matrix:



Interpretation: 63% predicted values are correctly classified with 37% misclassification rate by KKN classifier.

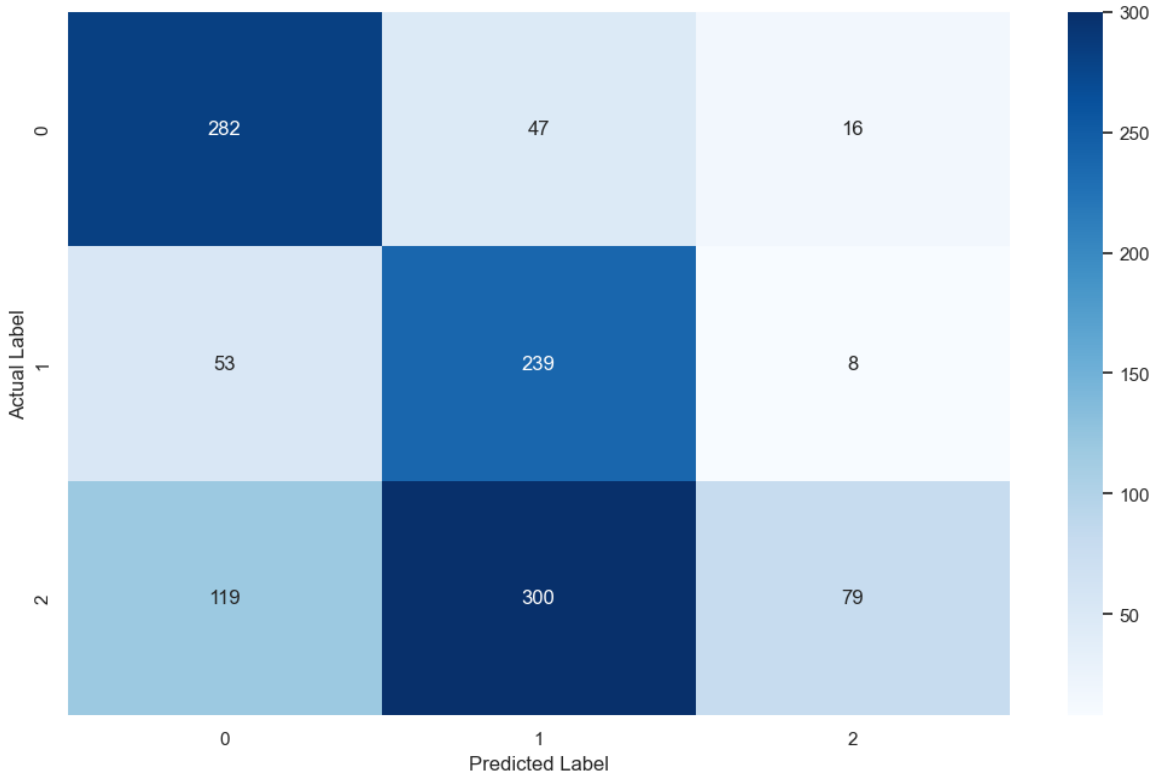
Naive Bayes Classifier

Naive Bayes models are a group of extremely fast and simple classification algorithms that are often suitable for very high-dimensional datasets.

Classification Report:

Precision	Recall	F1-Score	Accuracy	Roc Curve
0.60	0.59	0.50	0.52	0.89

Confusion Matrix:



Interpretation: 52% predicted values are correctly classified with 48% misclassification rate by KKN classifier.

Multinomial Logistic Regression

Multinomial Logistic Regression Equation:-

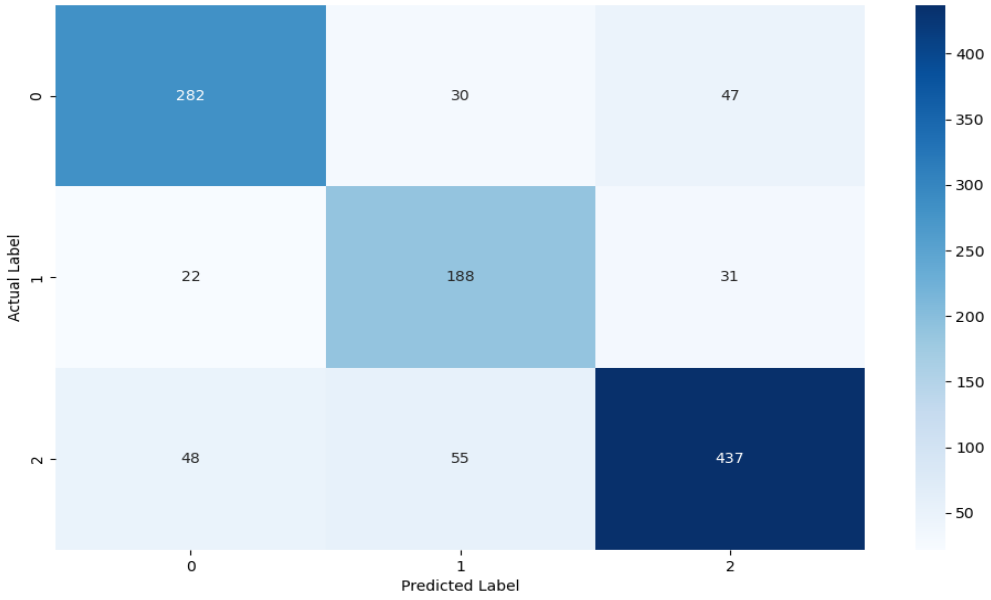
The log odds equation is given byLn{P[Job_Type=”data engineer”] / P[Job_Type=”data analyst”]}=

1.786788e+12+(-94.110373*Salary)+(-0.2546847520*No_of_Skills)+(1.720429*Review)+(0.716845782*Rating)+(-34800855*Date_Since_Posted)+(-0.265349959*Revenue)+(-0.26275013816*python)+(0.160305605597*sql)+(0.305291823962*machine_learning)+(0.155113421359*r)+(-0.16507235803*Hadoop)+(0.605284778757*tableau)+(0.44691275454*sas)+(-0.141644143446*spark)+(-0.500249357005*Java+(0.524122*CA)+(-0.14146969831*other)

Ln{P[Job_Type=”data scientist”] / P[Job_Type=”data analyst”]}=

1.786788e+10+(3.743362*Salary)+(-348171081*No_of_Skills)+(3.977626*Review)+(0.106825570*Rating)+(0.41985682*Date_Since_Posted)+(-0.207657904*Revenue)+(0.4124702293*python)+(-0.123658502786*sql)+(0.546522462413*machine_learning)+(0.159964645875*r)+(-0.12063306122*Hadoop)+(0.128535561870*tableau)+(0.237874498345*sas)+(-0.12330515878*spark)+(-0.31270734742*Java+(-0.66453337159*other)

Confusion Matrix:

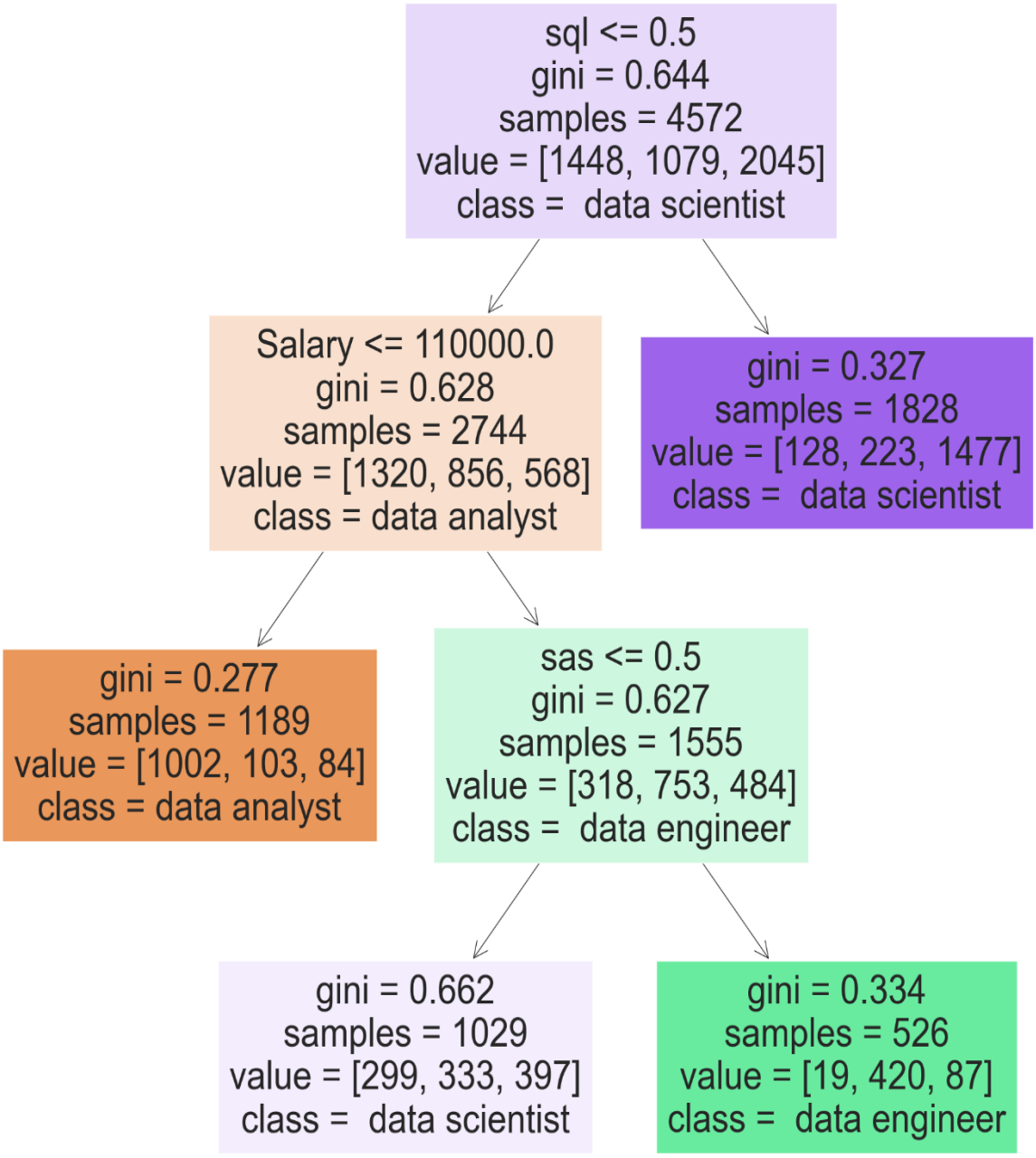


Interpretation: 79% predicted values are correctly classified with 21% misclassification rate by Multinomial Logistic Regression classifier.

Decision Tree Classifier

Decision tree induction is the learning of decision trees from class-labeled training tuples. From the decision tree classifier summary, Variable importance are {machine learning, Salary, Number of Skills, spark, python, r, hadoop, java, Other_states, sas}.

Decision Tree:



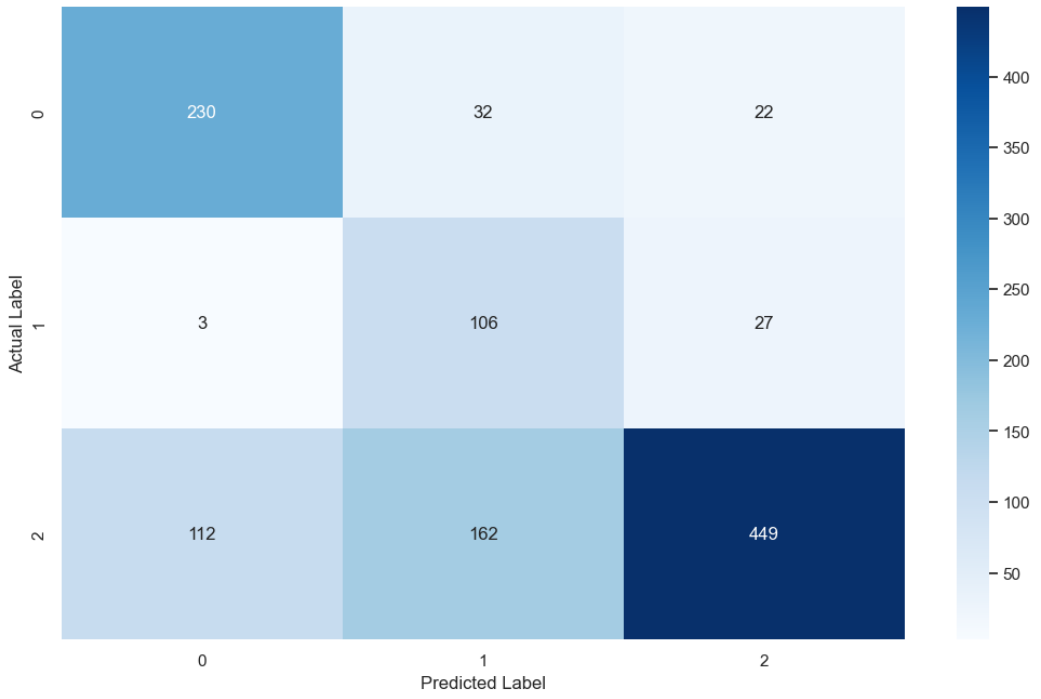
Cost complexity pruning:

Decision trees can easily overfit. One way to avoid it is to limit the growth of trees by setting constraints. We can limit parameters like *max_depth*, *min_samples*, etc. This helps to improve test accuracy and get a better model.

Classification Report

Precision	Recall	F1-Score	Accuracy	Roc Curve
0.70	0.69	0.69	0.70	0.85

Confusion Matrix:



Interpretation: 70% predicted values are correctly classified with 30% misclassification rate by Decision Tree classifier.

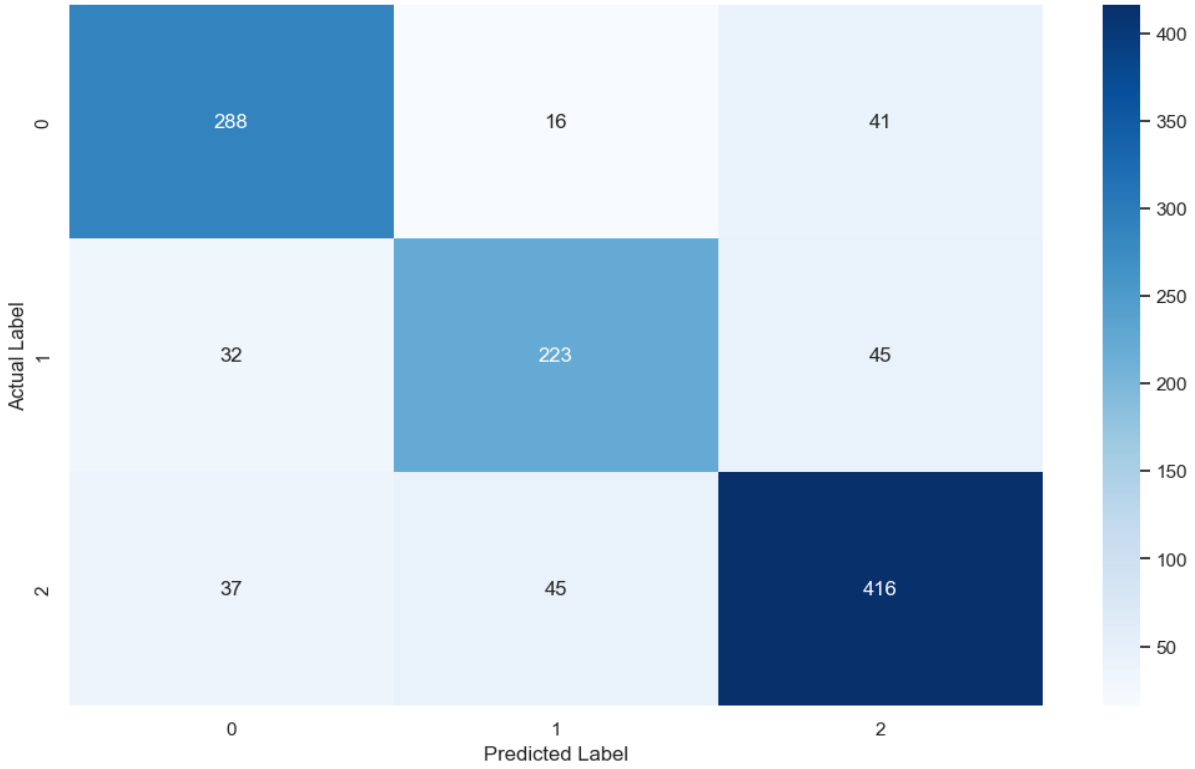
Random Forest Classifier

Random forest is another flexible supervised machine learning algorithm used for both classification and regression purposes. The "forest" references a collection of uncorrelated decision trees, which are then merged to reduce variance and create more accurate data predictions.

Classification Report:

Precision	Recall	F1-Score	Accuracy	Roc Curve
0.81	0.80	0.81	0.81	0.93

Confusion Matrix:



Interpretation: 81% predicted values are correctly classified with 19% misclassification rate by Random Forest classifier.

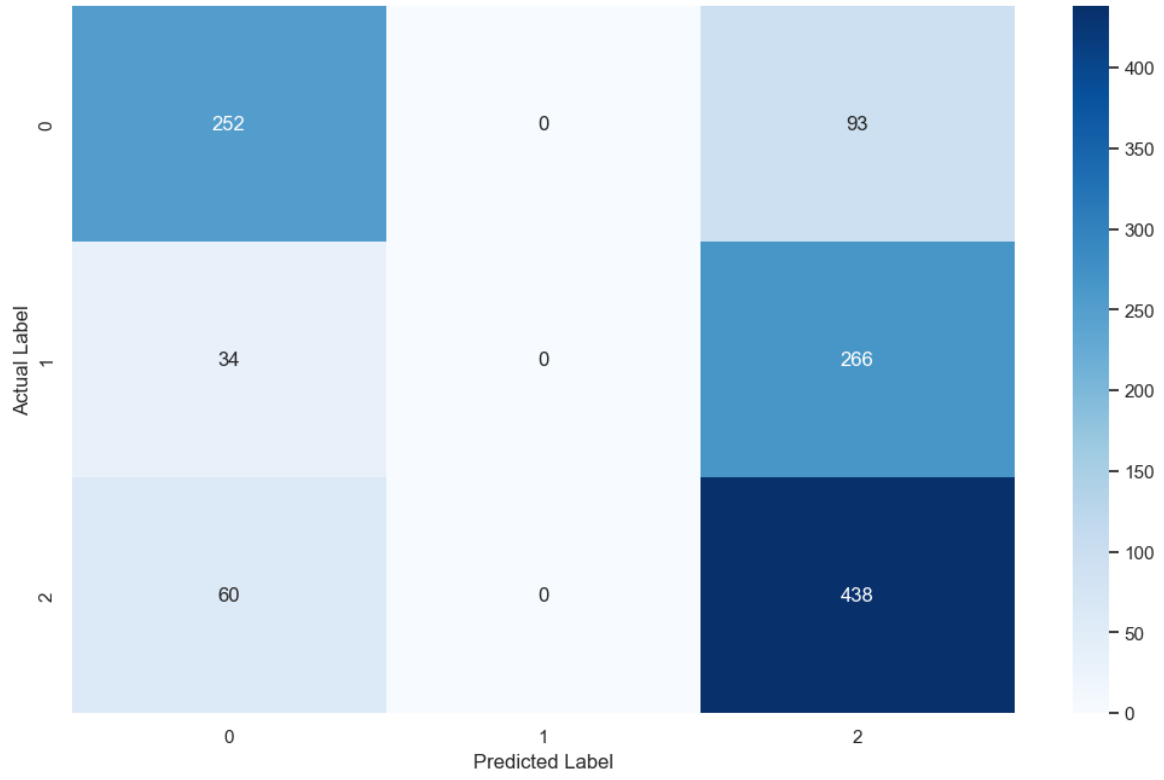
SUPPORT VECTOR MACHINE

In machine learning, SVM is used to classify data by finding the optimal decision boundary that maximally separates different classes. It aims to find the best hyperplane that maximizes the margin between support vectors, enabling effective classification even in complex, non-linear scenarios.

Classification Report:

Precision	Recall	F1-Score	Accuracy	Roc Curve
0.43	0.54	0.47	0.60	0.91

Confusion Matrix:

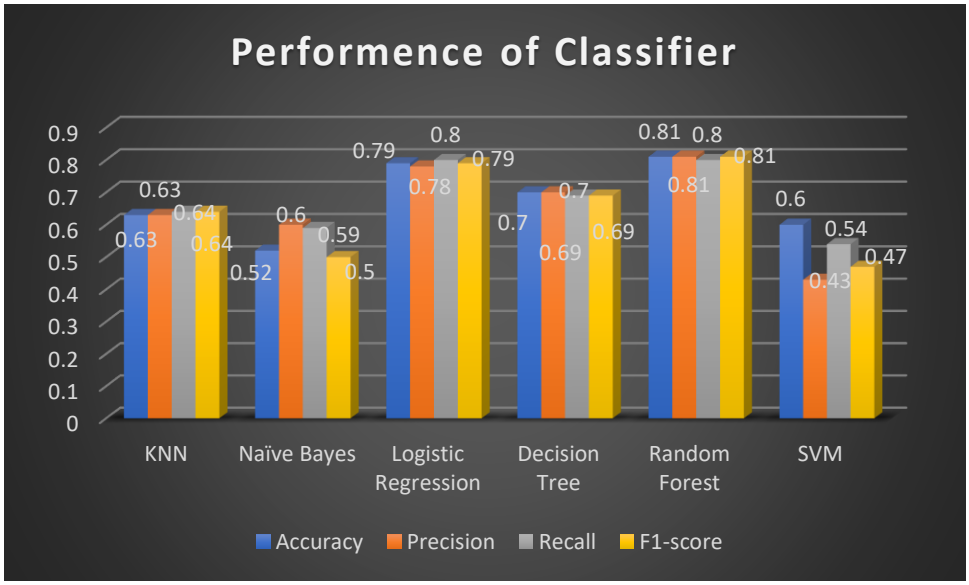


Interpretation: 60% predicted values are correctly classified with 40% misclassification rate by Support Vector Machine classifier.

Classification Report

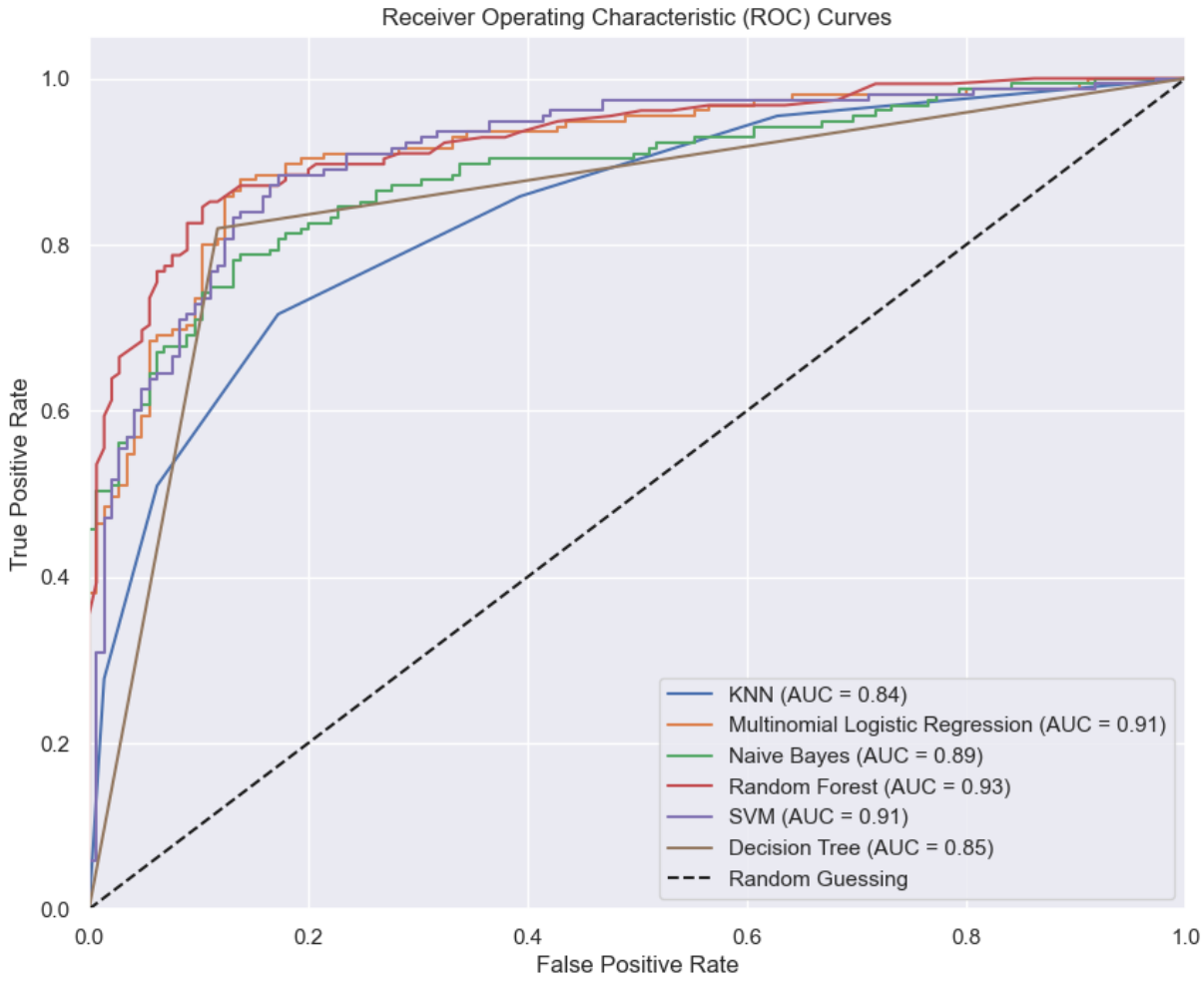
Classifiers	Accuracy	Precision	Recall	F1-score
KNN	**0.63**	**0.63**	**0.64**	**0.64**
Naïve Bayes	**0.52**	**0.60**	**0.59**	**0.50**
Logistic Regression	**0.79**	**0.78**	**0.80**	**0.79**
Decision Tree	**0.70**	**0.70**	**0.69**	**0.69**
Random Forest	**0.81**	**0.81**	**0.80**	**0.81**
SVM	**0.60**	**0.43**	**0.54**	**0.47**

Classification Report:



Interpretation: From above we conclude that Random Forest model gives highest (81%) accuracy of classification, were as logistic regression gives 79% accuracy.

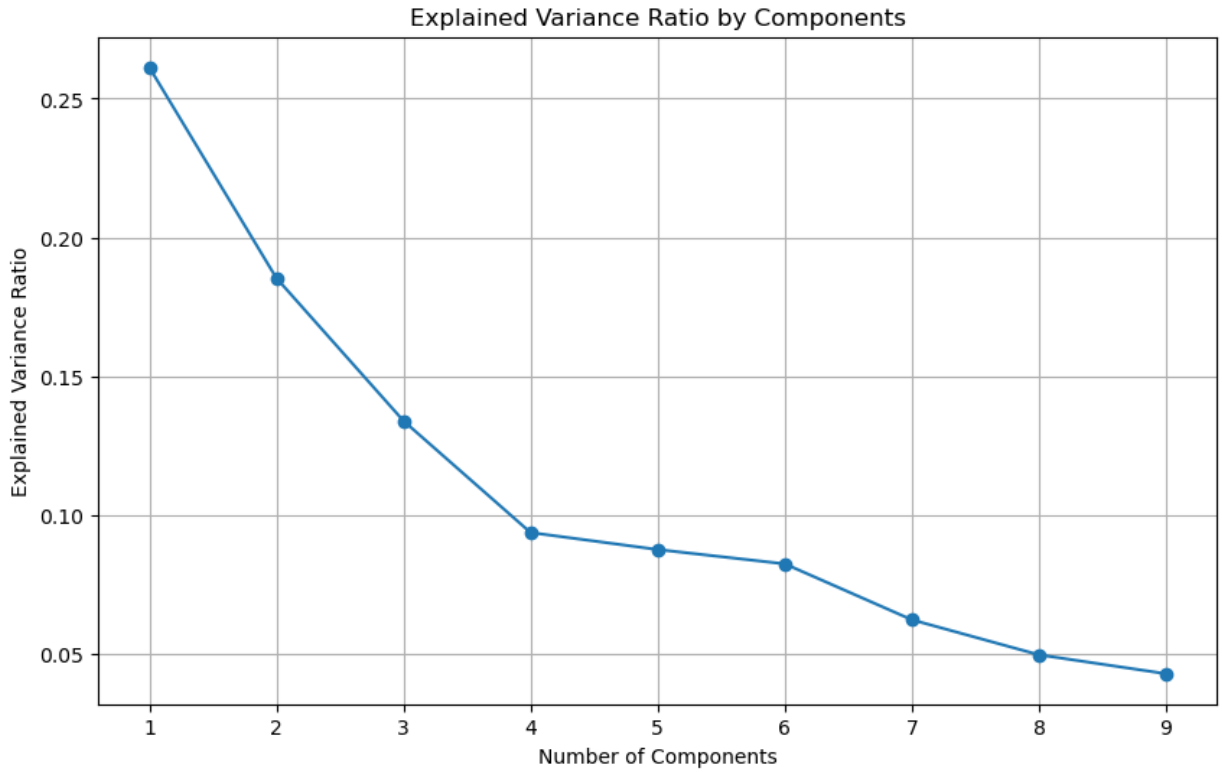
Receiver Operating Characteristic



Conclusion:

Random forest model gives highest accuracy **(93%)** of classification, whereas **logistic regression** gives **(91%)** accuracy.

PRINCIPLE COMPONENTS ANALYSIS



python	SQL	Machine learning	R	Hadoop	tableau	SAS	spark	java
0.2613	0.1854	0.134	0.0938	0.0877	0.0826	0.0624	0.0498	0.043

Conclusion:

We extracted 4 components from 9 components which covers maximum variation within dataset. This conclude that **SQL, Python, R,** and **Machine learning** are most preferable software for job **Data Analyst, Data Engineer** and **Data Scientist.**

❖ CONCLUSION

- From Visualization and our significant model, we can suggest that the most important skills for the candidates are **SAS, SQL, Python, R,** and **Machine learning** so that they can acquire them and get a suitable job.

- We build all **Supervised Machine learning algorithms** such as **KNN Classifier, Naive Bayes Classifier, Decision Tree Classifier, Random Forest Classifier,** and **Multinomial Logistic regression** and checked their accuracy and we get the best-supervised algorithm model. **i.e. Random Forest Classifier and its accuracy is 81%.**

- We checked the significant factors are **Salary, No. of_Skills, Reviews, Rating, Date_Since_Posted, Revenue, Employees** and **location** doesn't affect our target variable.

- We find out that the types of Job candidates prefer the most among the types **Data Scientist, Data Analyst,** and **Data Engineer is Data Scientist.**

- We build a recommendation system that will help candidates to get suitable jobs. From **Exploratory Data Analysis** we suggest to candidate important skills. We checked significant factors for our target variable. Also build different supervised learning models and conclude that from our model the **best model is Random Forest Classifier model.** Here from the sensitivity, Candidates will prefer **Data Scientist** jobs over other job types**.**

FUTURE WORK

- Extend the work and build a web service interface.
- Develop data skill vocabulary by exploring job descriptions instead of a pre-defined list of words.
- In this proposed work, there is only job recommendations and skill suggestion for IT jobs. It can be improved by suggesting jobs and skills for the Non–IT jobs. In the future, some can find a better choice to find similarity than a cosine similarity. It makes the recommendation more accurate.

LIMITATIONS

- In this project, we only consider three types of jobs.
- We only consider the Indeed job dataset. But we can build this model on other Job site datasets.

REFERENCES

[1] DATA MINING Concept and Techniques, Jiawei Han: Michline Kamber: Jian Pei

[2] Enhanced Job Recommendation System Shivraj Hulbatte[1], Amit Wabale[2], Suraj Patil , Nikhilkumar Sathe[4]

[3] Job Recommendation System Using Machine Learning And Natural Language Processing **JEEVANKRISHNA**

[4] JOB RECOMMENDATION SYSTEM BASED ON SKILL SETS G. Mahalakshmi[1], A. Arun Kumar[2], B.Senthilnayaki[1], J.Duraimurugan[1.]

[5] www.kaggle.com/datasets/elroyggj/indeed-dataset-data-scientist-analyst engineer.

APPENDIX

1)Normality

```
from scipy import stats
data=df[['Salary', 'No_of_Skills', 'Reviews',' Rating', 'Date_Since_Posted', 'Revenue',' Employees']]
k2, p = stats.normaltest(data)
alpha = 1e-3
if p.any() < alpha:  # null hypothesis: x comes from a normal distribution
   print("The null hypothesis can be rejected")
else:
   print("The null hypothesis cannot be rejected")
sns.heatmap(data.corr(),annot=True)
data.corr()

## Chi-Square Test
import pandas as pd
from scipy.stats import chi2_contingency

# Create DataFrame
df = pd.DataFrame(data)

# Create categories for No_of_Skills
bins = [0, 3, 6, 10]  # Bins: 1-3, 4-6, 7-10
labels = ['1-3', '4-6', '7-10']
df['No_of_Skills_Category'] = pd.cut(df['No_of_Skills'], bins=bins, labels=labels, include_lowest=True)

# Create contingency table
contingency_table = pd.crosstab(df['Salary'], df['No_of_Skills_Category'])

# Perform chi-square test
chi2, p, dof, expected = chi2_contingency(contingency_table)
```

```
    print("Chi-Square Statistic:", chi2)
    print("p-value:", p)
    print("Degrees of Freedom:", dof)
    print("Expected Frequencies:")
    print(pd.DataFrame(expected, index=contingency_table.index,
    columns=contingency_table.columns))

    ## Import File
    import pandas as pd
    import numpy as np
    import seaborn as sns
    import matplotlib.pyplot as plt
    from scipy import stats
    import warnings
    warnings.filterwarnings(action="ignore")
    Job_data=pd.read_csv("C:\\Users\\shubh\\Downloads\\job_data2.csv")
    Job_data.head()

## Data Preprocessing
 Job_data.shape
null=Job_data.isnull()
count_missing=null.count()
print(count_missing)
Job_data.describe()

##Data Validation
from sklearn.model_selection import train_test_split
Training_Data=Job_data.drop("Job_type",axis=1)
testing_Data=Job_data["Job_type"]
X_train,X_test,y_train,y_test=train_test_split(Training_Data,testing_Data,test_size=0.2)

## Data Balancing(over_sampling) by using SMOTE technique
from imblearn.over_sampling import SMOTE
print(X_train.shape,y_train.shape)
```

```
print(X_test.shape,y_test.shape)
print("before oversampling,count of lables'0':{}".format(sum(y_train==0)))
print("before oversampling,count of lables'1':{}".format(sum(y_train==1)))
print("before oversampling,count of lables'2':{}".format(sum(y_train==2)))
sm=SMOTE(random_state=30)
X_train_res, y_train_res= sm.fit_resample(X_train,y_train.ravel())
print("After oversampling,count of lables'0':{}".format(sum(y_train_res==0)))
print("After oversampling,count of lables'1':{}".format(sum(y_train_res==1)))
print("After oversampling,count of lables'2':{}".format(sum(y_train_res==2)))
```

Classification Technique:

1) KKN

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import
confusion_matrix,accuracy_score,mean_squared_error,classification_report,f1_score,Confusi
onMatrixDisplay
from sklearn import metrics
k=KNeighborsClassifier(n_neighbors=5)   # initializing model.
k.fit(X_train_res,y_train_res)   # fitting model
pred=k.predict(X_test.values)   # predicting model
ac1=metrics.accuracy_score(y_test,pred)
print("Accuracy from KNN =",ac1) cm1=confusion_matrix(y_test,pred)
sns.heatmap(cm1,annot=True,cmap='Blues', fmt='g')
plt.tight_layout()
plt.xlabel("Predicted Label")
plt.ylabel("Actual Label")
plt.gcf().set_size_inches(10,6)
plt.show()
print(classification_report(y_test,pred))

# I) GridSearchCV: to improve the KNN model:

from sklearn.model_selection import GridSearchCV
parameters = {"n_neighbors": range(1, 50)}
```

```
gridsearch = GridSearchCV(KNeighborsClassifier(), parameters)
gridsearch.fit(X_train, y_train)
gridsearch.best_params_
y_pred=gridsearch.predict(X_test)
accuracy = accuracy_score(y_pred, y_test)
f1=f1_score(y_pred, y_test, average="weighted")
print("Accuracy:", accuracy)
print("F1 Score:", f1)

# II) GridSearchCV: to improve the KNN model :

from sklearn.model_selection import GridSearchCV
parameters = {"n_neighbors": range(1, 50), "weights": ["uniform", "distance"]}
gridsearch = GridSearchCV(KNeighborsClassifier(), parameters)
gridsearch.fit(X_train, y_train)
gridsearch.best_params_
y_pred=gridsearch.predict(X_test)
accuracy = accuracy_score(y_pred, y_test)
f1 = f1_score(y_pred, y_test, average="weighted")
print("Accuracy:", accuracy)
print("F1 Score:", f1)

2) Naïve Bayes
from sklearn.naive_bayes import GaussianNB
n=GaussianNB()
n.fit(X_train_res,y_train_res)
pred2=n.predict(X_test)
pred2
ac2=metrics.accuracy_score(y_test,pred2)
print("Accuracy of Naive Bayes =",ac2)
print(classification_report(y_test,pred2))
cm2=confusion_matrix(y_test,pred2)
sns.heatmap(cm2,annot=True,cmap='Blues', fmt='g')
plt.tight_layout()
```

```
plt.xlabel("Predicted Label")
plt.ylabel("Actual Label")
plt.gcf().set_size_inches(10,6)
plt.show()

3)Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
DT=DecisionTreeClassifier(max_leaf_nodes=5)
DT.fit(X_train_res,y_train_res)
pred4=DT.predict(X_test)
pred4
ac4=metrics.accuracy_score(y_test,pred4)
print("Accuracy of Decision Tree =",ac4)
print(classification_report(y_test,pred4))
from sklearn import tree
t=tree.export_text(DT)
print(t)

# Decision Tree Classifier prunning :
path = DT.cost_complexity_pruning_path(X_train, y_train)
ccp_alphas, impurities = path.ccp_alphas, path.impurities
fig, ax = plt.subplots()
ax.plot(ccp_alphas[:-1], impurities[:-1], marker="o", drawstyle="steps-post")
ax.set_xlabel("effective alpha")
ax.set_ylabel("total impurity of leaves")
ax.set_title("Total Impurity vs effective alpha for training set")
DTs = []
for ccp_alpha in ccp_alphas:
    DT = DecisionTreeClassifier(random_state=0, ccp_alpha=ccp_alpha)
    DT.fit(X_train, y_train)
    DTs.append(DT)
print(
    "Number of nodes in the last tree is: {} with ccp_alpha: {}".format(
        DTs[-1].tree_.node_count, ccp_alphas[-1]
```

```
    )
)
DTs = DTs[:-1]
ccp_alphas = ccp_alphas[:-1]

node_counts = [DT.tree_.node_count for DT in DTs]
depth = [DT.tree_.max_depth for DT in DTs]
fig, ax = plt.subplots(2, 1)
ax[0].plot(ccp_alphas, node_counts, marker="o", drawstyle="steps-post")
ax[0].set_xlabel("alpha")
ax[0].set_ylabel("number of nodes")
ax[0].set_title("Number of nodes vs alpha")
ax[1].plot(ccp_alphas, depth, marker="o", drawstyle="steps-post")
ax[1].set_xlabel("alpha")
ax[1].set_ylabel("depth of tree")
ax[1].set_title("Depth vs alpha")
fig.tight_layout()
train_scores = [DT.score(X_train, y_train) for DT in DTs]
test_scores = [DT.score(X_test, y_test) for DT in DTs]
fig, ax = plt.subplots()
ax.set_xlabel("alpha")
ax.set_ylabel("accuracy")
ax.set_title("Accuracy vs alpha for training and testing sets")
ax.plot(ccp_alphas, train_scores, marker="o", label="train", drawstyle="steps-post")
ax.plot(ccp_alphas, test_scores, marker="o", label="test", drawstyle="steps-post")
ax.legend()
plt.show()
DTs = [ ]
for ccp_alpha in ccp_alphas:
    DT = tree.DecisionTreeClassifier(random_state=0, ccp_alpha=ccp_alpha)
    DT.fit(X_train, y_train)
    DTs.append(DT)
DTs = DTs[:-1]
ccp_alphas = ccp_alphas[:-1]
```

```
node_counts = [DT.tree_.node_count for DT in DTs]
depth = [DT.tree_.max_depth for DT in DTs]
plt.scatter(ccp_alphas,node_counts)
plt.scatter(ccp_alphas,depth)
plt.plot(ccp_alphas,node_counts,label='no of nodes',drawstyle="steps-post")
plt.plot(ccp_alphas,depth,label='depth',drawstyle="steps-post")
plt.legend( )
plt.show( )
train_acc = [ ]
test_acc = [ ]
for c in DTs:
    y_train_pred = c.predict(X_train)
    y_test_pred = c.predict(X_test)
    train_acc.append(accuracy_score(y_train_pred,y_train))
    test_acc.append(accuracy_score(y_test_pred,y_test))
plt.scatter(ccp_alphas,train_acc)
plt.scatter(ccp_alphas,test_acc)
plt.plot(ccp_alphas,train_acc,label='train_accuracy',drawstyle="steps-post")
plt.plot(ccp_alphas,test_acc,label='test_accuracy',drawstyle="steps-post")
plt.legend()
plt.title('Accuracy vs alpha')
plt.show()
def plot_confusionmatrix(y_test,y_pred,dom):
    print(f'{dom} Confusion matrix')
    cf = confusion_matrix(y_test,y_pred)
    sns.heatmap(cf,annot=True,cmap='Blues', fmt='g')
    plt.tight_layout()
    plt.xlabel("Predicted Label")
    plt.ylabel("Actual Label")
    plt.gcf().set_size_inches(10,6)
    plt.show()
DT_ = tree.DecisionTreeClassifier(random_state=0,ccp_alpha=0.020)
DT_.fit(X_train,y_train)
y_train_pred = DT_.predict(X_train)
```

```
y_test_pred = DT_.predict(X_test)
print(f'Train score {accuracy_score(y_train_pred,y_train)}')
print(f'Test score {accuracy_score(y_test_pred,y_test)}')
plot_confusionmatrix(y_train_pred,y_train,dom='Train')
plot_confusionmatrix(y_test_pred,y_test,dom='Test')
features = Job_data.columns
classes = ['data analyst',' data engineer',' data scientist']
tree.plot_tree(DT_,feature_names=features,class_names=classes,filled=True)
plt.show()
```

4) Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
r=RandomForestClassifier(n_estimators=10)
r.fit(X_train_res,y_train_res)
pred5=r.predict(X_test)
pred5
ac5=metrics.accuracy_score(y_test,pred5)
print("Accuracy of Random Forest =",ac5)
print(classification_report(y_test,pred5))
cm5=confusion_matrix(y_test,pred5)
sns.heatmap(cm5,annot=True,cmap='Blues', fmt='g')
plt.tight_layout()
plt.xlabel("Predicted Label")
plt.ylabel("Actual Label")
plt.gcf().set_size_inches(10,6)
plt.show()
```

5) Support Vector Machine Classifier

```
from sklearn.svm import SVC
svm=SVC()
svm.fit(X_train,y_train)
pred6=svm.predict(X_test)
pred6
ac6=metrics.accuracy_score(y_test,pred6)
```

```
print("Accuracy from Support Vector Machine =",ac6)
print(classification_report(y_test,pred6))
cm6=confusion_matrix(y_test,pred6)
print(cm6)
cm6=confusion_matrix(y_test,pred6)
sns.heatmap(cm6,annot=True,cmap='Blues', fmt='g')
plt.tight_layout()
plt.xlabel("Predicted Label")
plt.ylabel("Actual Label")
plt.gcf().set_size_inches(10,6)
plt.show()

## Correlation Heatmap
data.corr()
plt.figure(figsize=(10,8))
sns.heatmap(data.corr(),annot=True, cmap="viridis")



## ROC
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import roc_curve, auc

# Generate some example data
X, y = make_classification(n_samples=1000, n_features=20, n_classes=2, random_state=42)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize all classifiers
classifiers = {
    "KNN": KNeighborsClassifier(),
    "Multinomial Logistic Regression": LogisticRegression(),
    "Naive Bayes": GaussianNB(),
    "Random Forest": RandomForestClassifier(),
    "SVM": SVC(probability=True),
    "Decision Tree": DecisionTreeClassifier()
}

# Train all classifiers and compute ROC curves
plt.figure(figsize=(10, 8))
for name, clf in classifiers.items():
    clf.fit(X_train, y_train)
    y_score = clf.predict_proba(X_test)[:, 1]
    fpr, tpr, _ = roc_curve(y_test, y_score)
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr, tpr, label=f'{name} (AUC = {roc_auc:.2f})')

# Plot ROC curve for each classifier
plt.plot([0, 1], [0, 1], linestyle='--', color='k', label='Random Guessing')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curves')
plt.legend(loc='lower right')
```

```
plt.show()
```

R-Code

```
# Multinomial Logistic Regression

Data=read.csv(file.choose(T),header=T)
head(Data)
Data$Job_type=as.factor(Data$Job_type)
str(Data)

# Train and test dataset:
id=sample(2,nrow(Data),replace=T,prob=c(0.8,0.2))
train=Data[id==1,]
nrow(train)
test=Data[id==2,]
nrow(test)
head(test)
# Develop Multinomial Logistic Regression Model :
library(nnet)
train$Job_type<-relevel(train$Job_type,ref = "1")
model<-multinom(Job_type ~.,train)
nrow(model)
summary(model)
# prediction of testing data
p1<-predict(model,test)
head(p1)
tab1=table(p1,test$Job_type)
tab2=c(282,30,47,22,188,31,48,55,437)
tab3=matrix(tab2,nrow=3,ncol=3,byrow=FALSE,list(c("0","1","2"),c("0","1","2")))
tab3
Accuracy=sum(diag(tab3))/sum(tab3)
Accuracy
```

```
z=summary(model)$coefficients/summary(model)$standard.errors
z
p=(1-pnorm(abs(z),0,1))*2
p
n=table(train$Job_type)
n
```