Section - 1

**project overview :** In this project, we developed a classification-based machine learning model using Python. We will be worked with a real dataset that contains at least 2000 instances, and we will perform data preprocessing to ensure that the dataset is suitable for modeling. We also conducted exploratory data analysis to identify any patterns or trends in the data. We have used several classification models, including naive Bayes, KNN, decision tree, logistic regression, and support vector machines, to classify the data based on specific features. We compared the models based on their predictive accuracy to determine the best model for the dataset. The project will be summarized in a report, including details of the dataset, data preprocessing, EDA, and model development, with relevant screenshots and plots.

Section - 2

**Dataset overview**

**Data Source with Valid URL**

<https://data.nashville.gov/General-Government/General-Government-Employees-Titles-and-Base-Annua/2hu7-5kjq>

**Description About Dataset**

This dataset contains 2543 total rows and 5 columns of information about Metro Nashville's general government employees, including their name, job title, department, employment status, and base annual salary. The dataset includes various job titles, ranging from police officers and sergeants to maintenance leaders, program specialists, and family development workers.

* **Name** : The name column lists the employees' full names.
* **Titles :** Title column provides their job titles, indicating the specific role they fulfill within their department.
* **Department :** The department column identifies the department that the employee belongs to, such as police, parks, public library, or juvenile court.
* **Employment Status :** The employment status column indicates whether the employee is full-time, part-time, or temporary.
* **Annual Salaries :** The annual salary column lists the base annual salary of each employee, providing an idea of their compensation for their job.

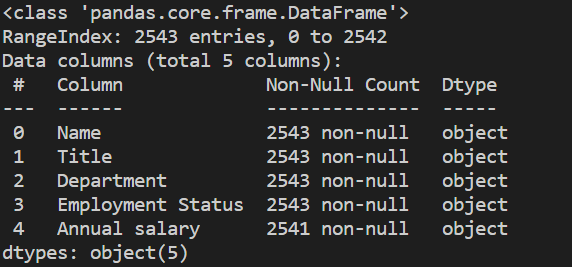
Overall, this dataset provides valuable information about the employees of Metro Nashville's general government and their respective job titles and salaries. It could be useful for conducting analyses on government salaries, workforce diversity, and other related topics.

Section - 3

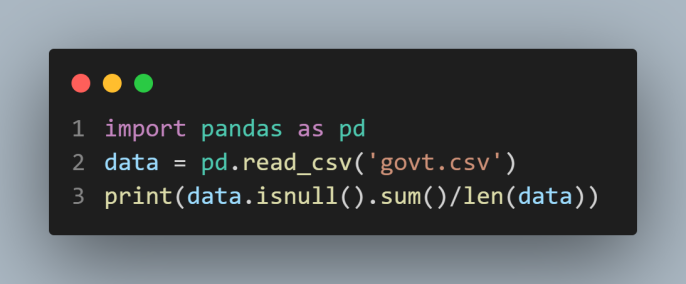
**Data Preprocessing and Exploratory data analysis**

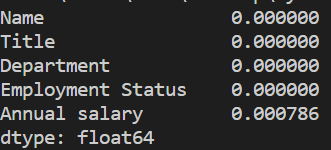
**Data Preprocessing:**



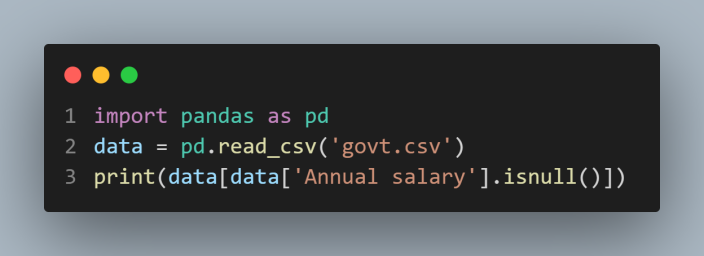


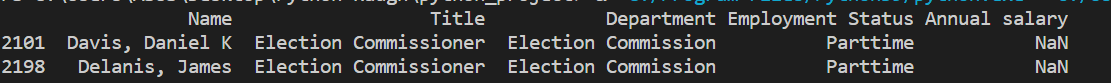
* First we load our .csv dataset file with pandas , and check the information of our dataset. It shows our features names , non-null value count and the data type from our dataset. Expect the Annual salary column , we have same amount of value in all other columns.



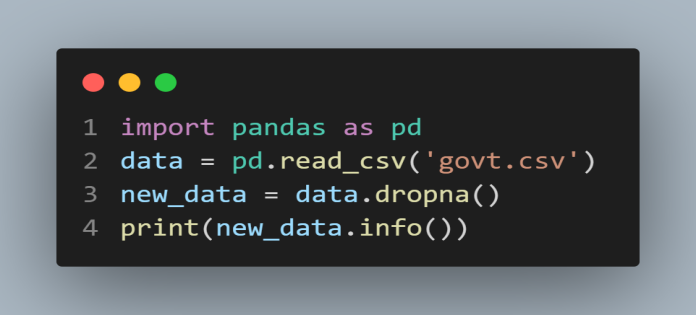


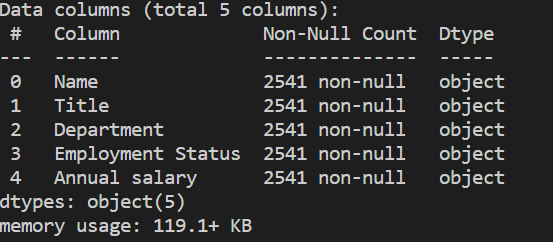
* There are few percentages of null data in ‘Annual salary’ column.





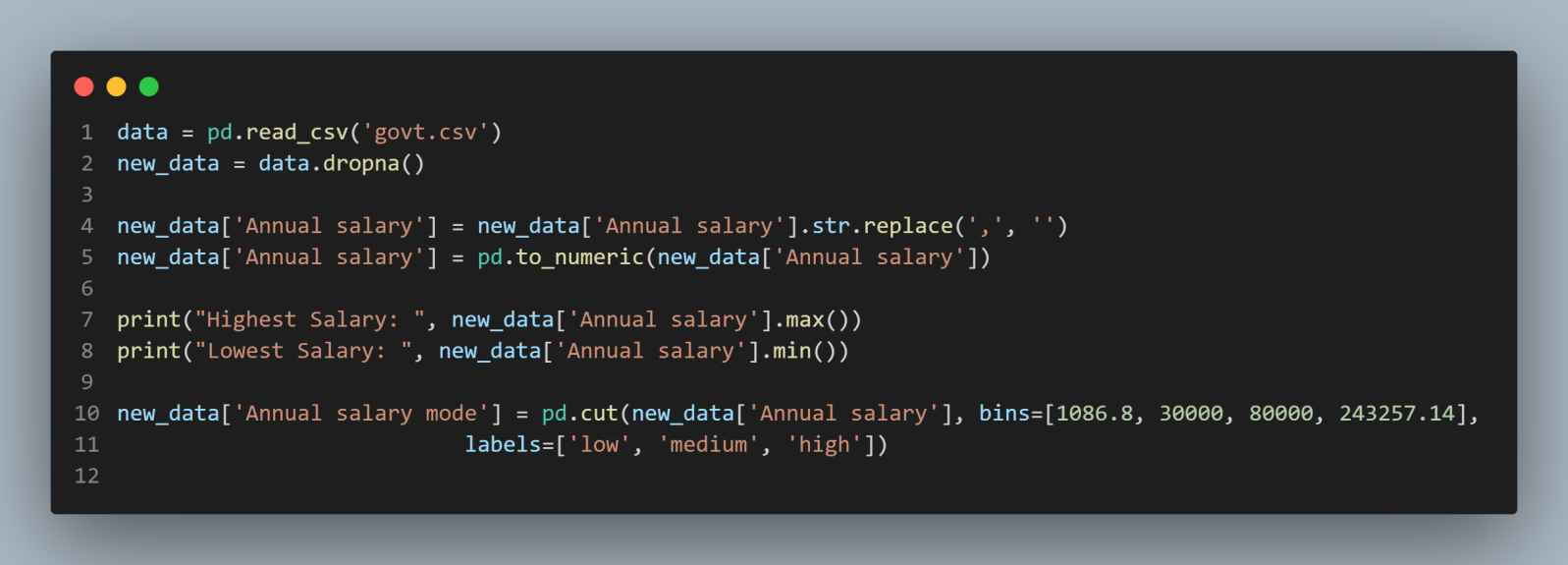
* Just drop these 2 rows of missing data.



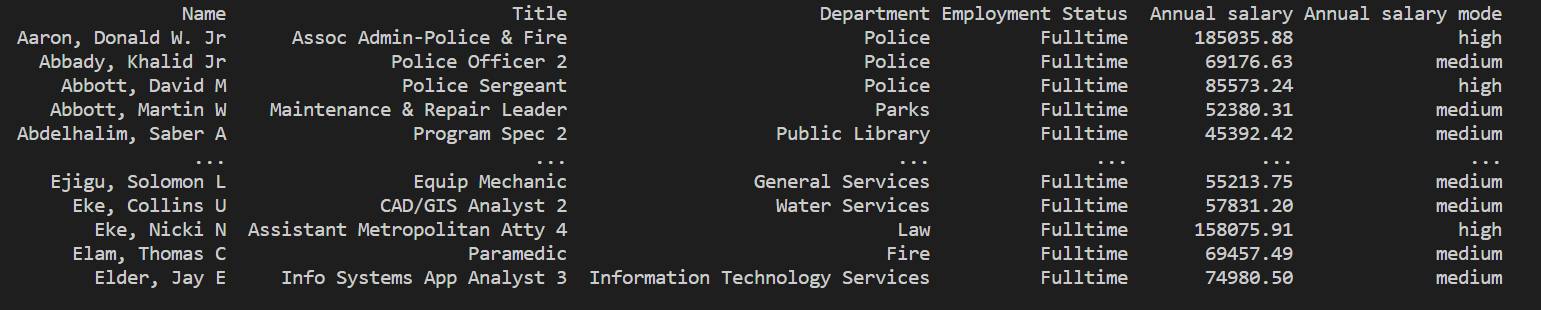


* Now, our data frame is fully clean, there is no missing value or null data.

**Exploratory data analysis:**

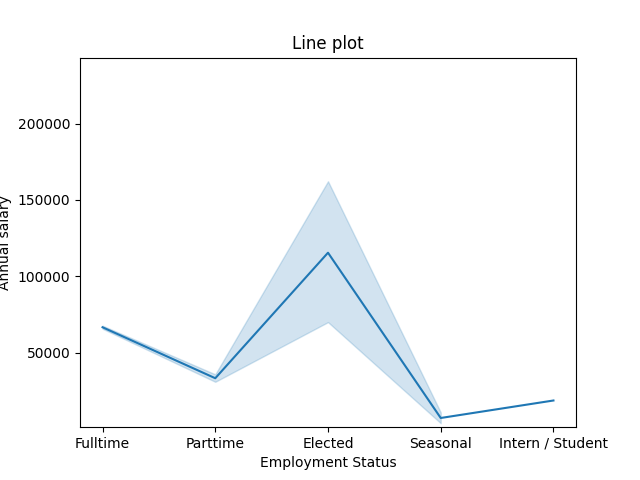


* For, better result and understand the data process ,we introduce a new column, ‘Annual salary mode’.
* In this new column there are 3 values ‘low’, ‘medium’ and ‘high’ which indicates the salary level of an employee.
* First we check the lowest and highest annual salary of an employee, then set a range 1086.8 to 30000 low range, 30000 to 80000 medium range and from 80000 to 243257.14 is high range.
* According to mention range the new columns value will be set automatically.



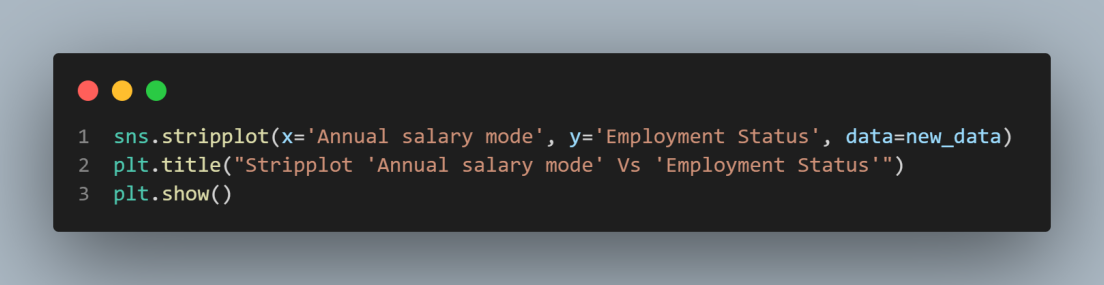
**Line Plot**

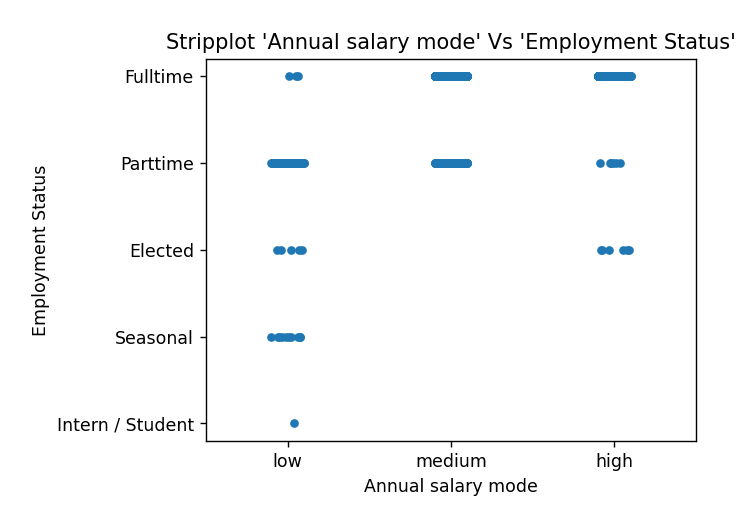


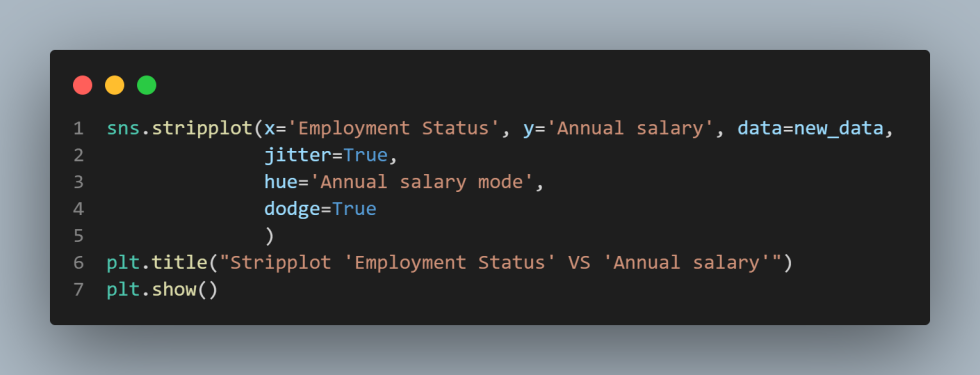


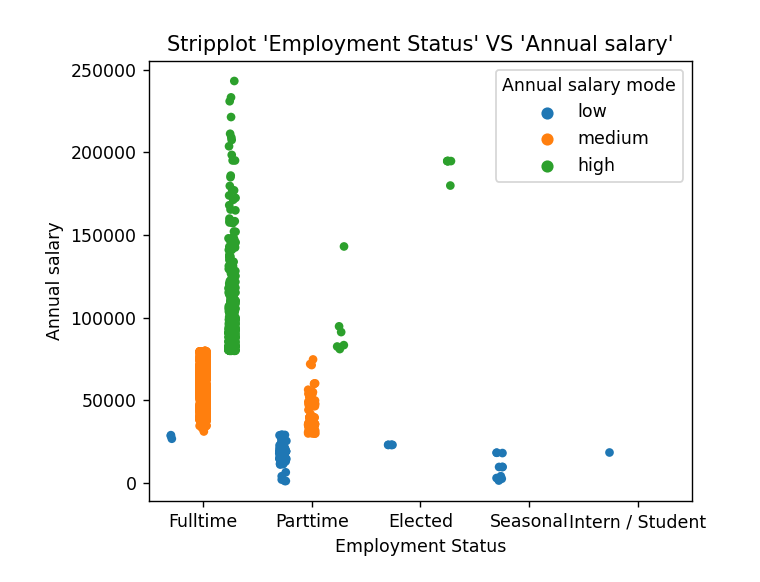
* This line plot shows that , the Elected employees are get maximum salary and seasonal employee gets the lowest value.

**Strip Plot**



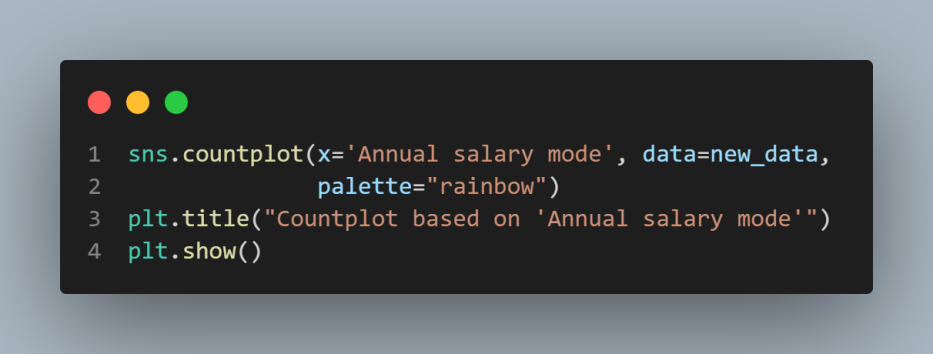


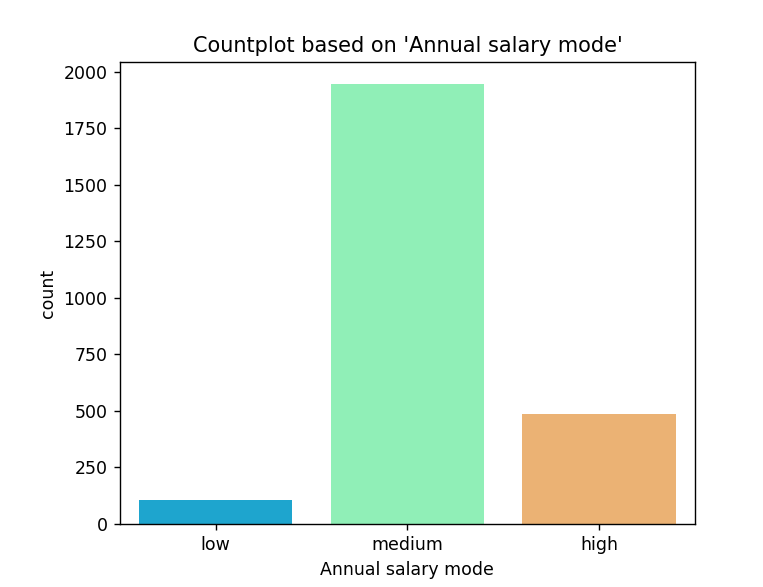




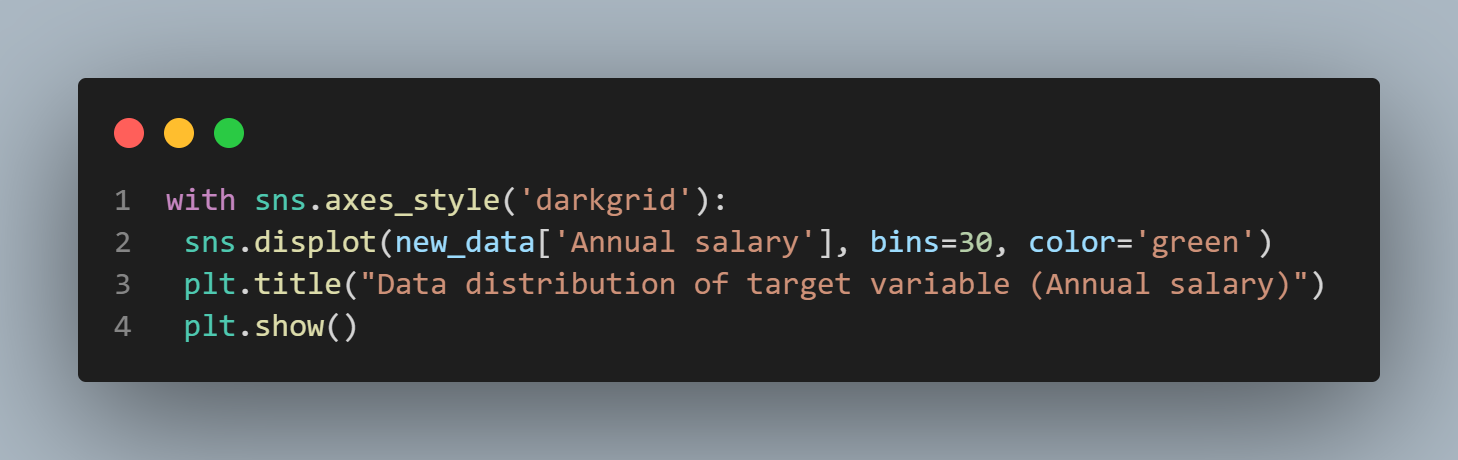
* This stripplot shows the annual salary based on the employment status.
* For understand more details, we separate a new section with Annual salary mode use as hue and true the dodge for plot separate section based on hue.
* As we can see from the graph, employee with full-time status got the highest salary, and part-time employee got most significant number of low salary.

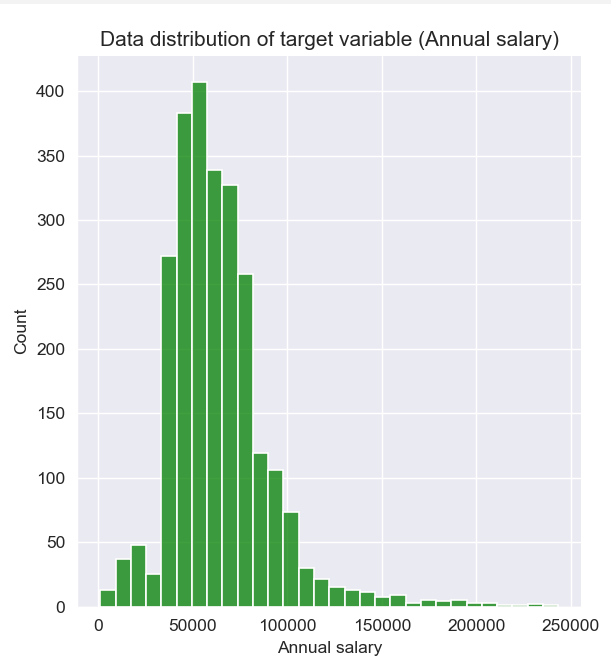
**Count Plot**

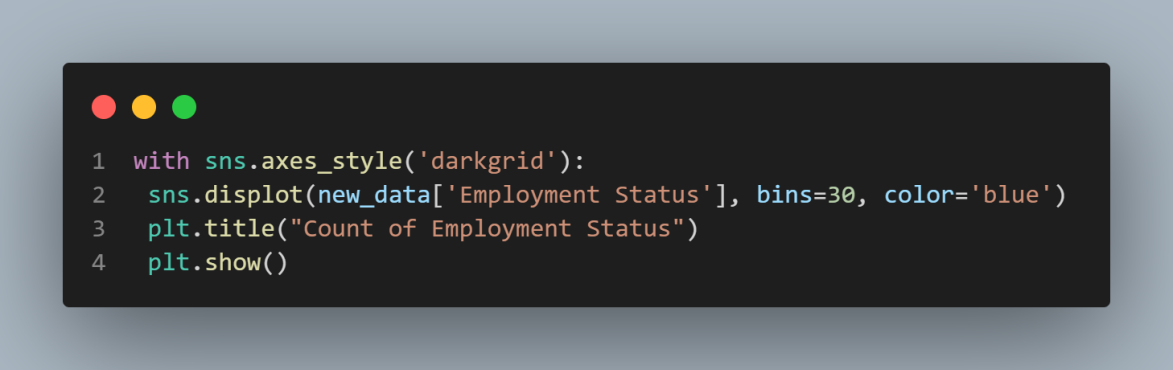


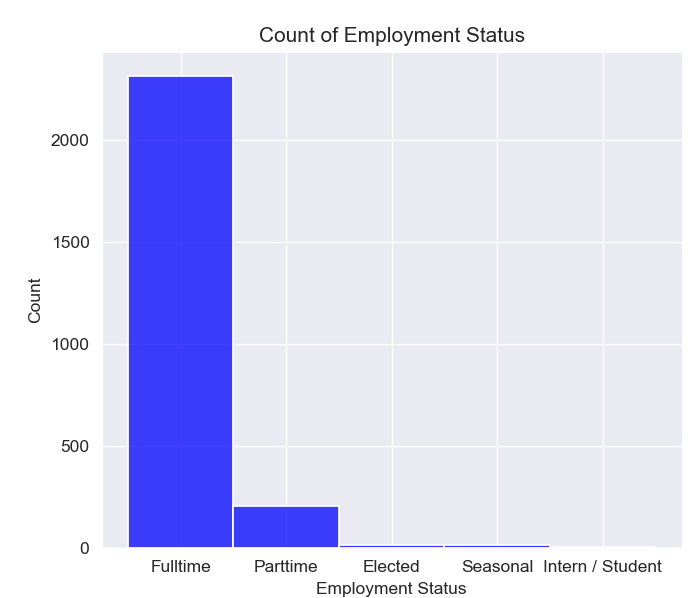


* Count plot , count range of salary, and it display that most of employee paid with medium range of salary .



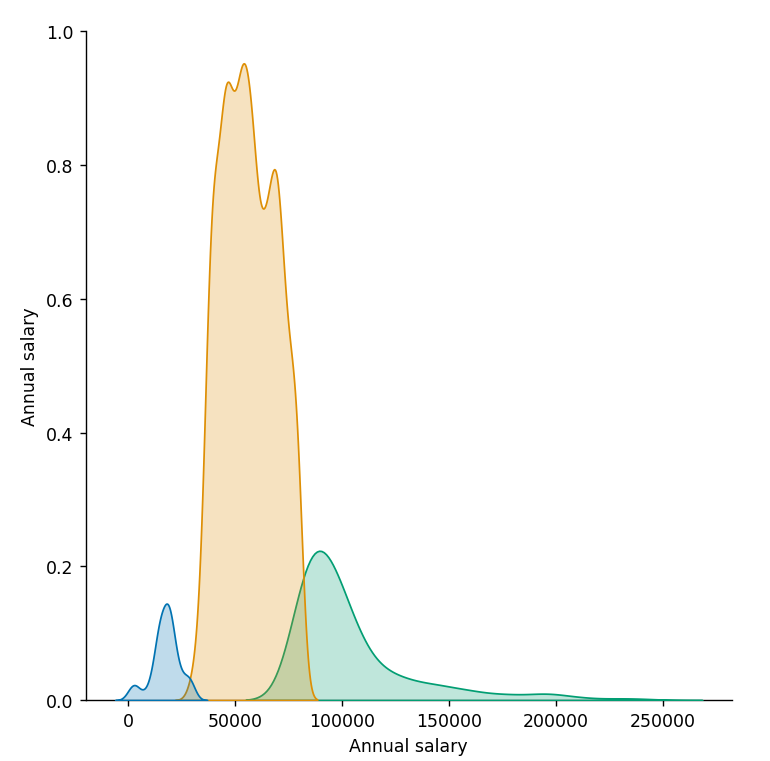






* Maximum number of full-time employees.





* Annual salary, which is our target variable, is correlated with itself with the value +1.
* In this pair plot , separate the section based on Annual salary mode using hue.

Section - 4

**Model Development**

**Development process SVM Model:**

After loading the dataset, we Pre-Preprocess our dataset by dropping rows with missing values in the target variable, and split the data into training and testing sets.

After that we have convert all categorical variables to numeric values using X = pd.get\_dummies(X) encoding in pandas.

Then we select the SVM algorithm model = SVC() as the machine learning model to train our dataset called “govt.csv”.

Then we train the SVM model using the training dataset with the fit method.

After that, We evaluate the model using the testing dataset with the predict method and calculate accuracy using the accuracy\_score function.

We calculated the F1 score using the f1\_score() function, passing in the actual test data labels (y\_test) and predicted labels (y\_pred), and specifying the average parameter to be 'weighted'.

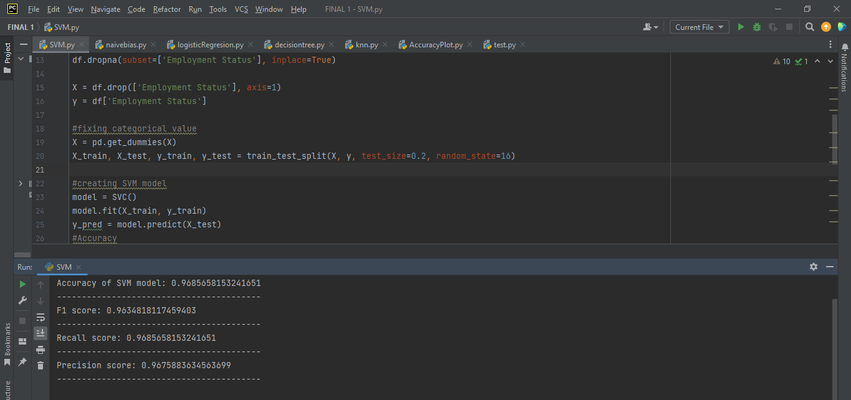
We calculated the recall score using the recall\_score() function, passing in the actual test data labels (y\_test) and predicted labels (y\_pred), and specifying the average parameter to be 'weighted'.

We calculated the precision score using the precision\_score() function, passing in the actual test data labels (y\_test) and predicted labels (y\_pred), and specifying the average parameter to be 'weighted' and zero\_division parameter to be 1.

**Code is shown below:**

**import** pandas **as** pd  
**from** sklearn.svm **import** SVC  
**from** sklearn.model\_selection **import** train\_test\_split  
**from** sklearn.metrics **import** accuracy\_score, f1\_score, recall\_score, precision\_score  
  
*# loading the dataset*df = pd.read\_csv(**"govt.csv"**)  
*# drop name column as it is not workable for measure accuracy*data = df.drop(**'Name'**, axis=1)  
df.dropna(subset=[**'Employment Status'**], inplace=**True**)  
  
X = df.drop([**'Employment Status'**], axis=1)  
y = df[**'Employment Status'**]  
  
*# fixing categorical value*X = pd.get\_dummies(X)  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=16)  
  
*# creating SVM model*model = SVC()  
model.fit(X\_train, y\_train)  
y\_pred = model.predict(X\_test)  
*# Accuracy*accuracy = accuracy\_score(y\_test, y\_pred)  
print(**"-----------------------------------------"**)  
print(**"Accuracy of SVM model:"**, accuracy)  
print(**"-----------------------------------------"**)  
*# f1\_score*f1 = f1\_score(y\_test, y\_pred, average=**'weighted'**)  
print(**'F1 score:'**, f1)  
print(**"-----------------------------------------"**)  
*# recall*recall = recall\_score(y\_test, y\_pred, average=**'weighted'**)  
print(**'Recall score:'**, recall)  
print(**"-----------------------------------------"**)  
*# precision*precision = precision\_score(y\_test, y\_pred, average=**'weighted'**, zero\_division=1)  
print(**'Precision score:'**, precision)  
print(**"-----------------------------------------"**)

**Screen shoot of Output:**



**Development process of Naïve bias Model:**

After loading the dataset, we Pre-Preprocess our dataset by dropping rows with missing values in the target variable, and split the data into training and testing sets.

After that we have convert all categorical variables to numeric values using X = pd.get\_dummies(X) encoding in pandas.

Then we select the model = GaussianNB() algorithm as the machine learning model to train our dataset called “govt.csv”.

Then we train the Navie bias model using the training dataset with the fit method.

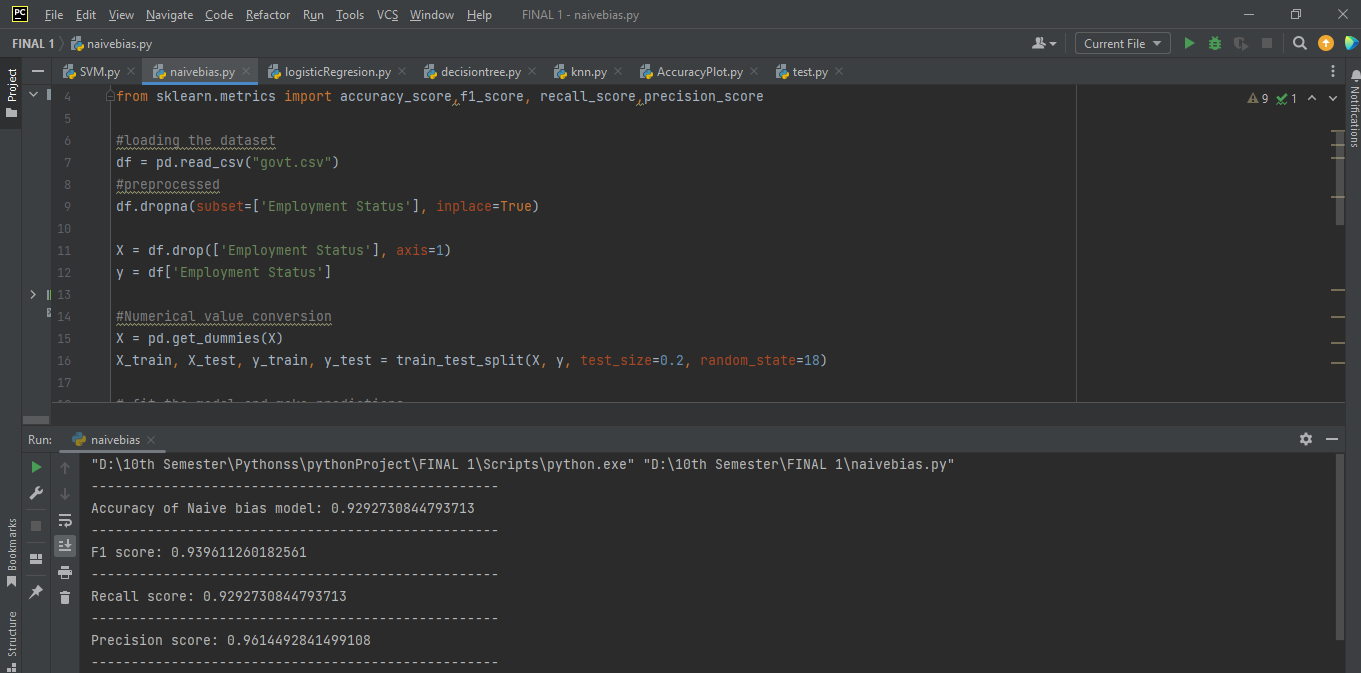
After that, We evaluate the model using the testing dataset with the predict method and calculate accuracy using the accuracy\_score function.

We calculated the F1 score using the f1\_score() function, passing in the actual test data labels (y\_test) and predicted labels (y\_pred), and specifying the average parameter to be 'weighted'.

We calculated the recall score using the recall\_score() function, passing in the actual test data labels (y\_test) and predicted labels (y\_pred), and specifying the average parameter to be 'weighted'.

We calculated the precision score using the precision\_score() function, passing in the actual test data labels (y\_test) and predicted labels (y\_pred), and specifying the average parameter to be 'weighted' and zero\_division parameter to be 1.

**Screen shoot of Output:**



**Development process KNN Model:**

After loading the dataset, we Pre-Preprocess our dataset by dropping rows with missing values in the target variable, and split the data into training and testing sets.

After that we have convert all categorical variables to numeric values using X = pd.get\_dummies(X) encoding in pandas.

Then we select the KNN algorithm model = KNeighborsClassifier() as the machine learning model to train our dataset called “govt.csv”.

Then we train the KNN model using the training dataset with the fit method.

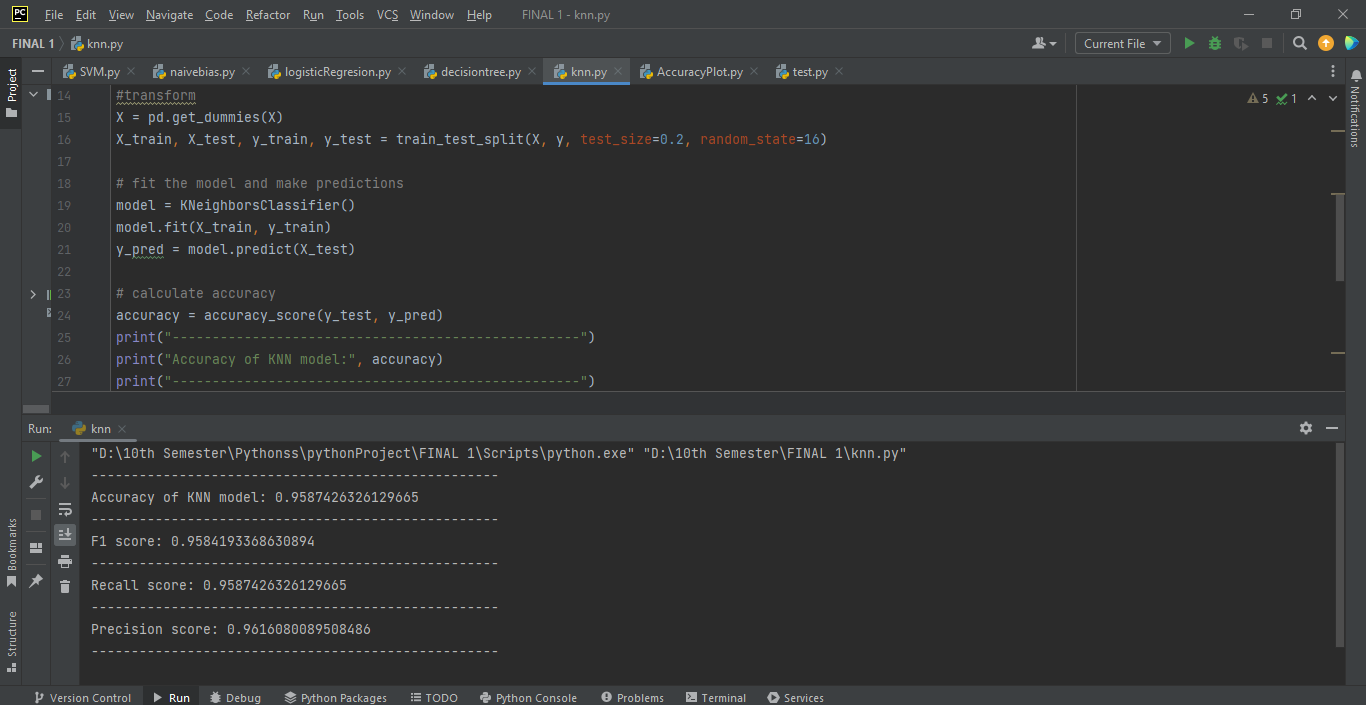
After that, We evaluate the model using the testing dataset with the predict method and calculate accuracy using the accuracy\_score function.

We calculated the F1 score using the f1\_score() function, passing in the actual test data labels (y\_test) and predicted labels (y\_pred), and specifying the average parameter to be 'weighted'.

We calculated the recall score using the recall\_score() function, passing in the actual test data labels (y\_test) and predicted labels (y\_pred), and specifying the average parameter to be 'weighted'.

We calculated the precision score using the precision\_score() function, passing in the actual test data labels (y\_test) and predicted labels (y\_pred), and specifying the average parameter to be 'weighted' and zero\_division parameter to be 1.

**Screen shoot of Output:**



**Development process Decision Tree Model:**

After loading the dataset, we Pre-Preprocess our dataset by dropping rows with missing values in the target variable, and split the data into training and testing sets.

After that we have convert all categorical variables to numeric values using X = pd.get\_dummies(X) encoding in pandas.

Then we select the Decision Tree algorithm dtc = DecisionTreeClassifier() as the machine learning model to train our dataset called “govt.csv”.

Then we train the Decision Tree model using the training dataset with the fit method.

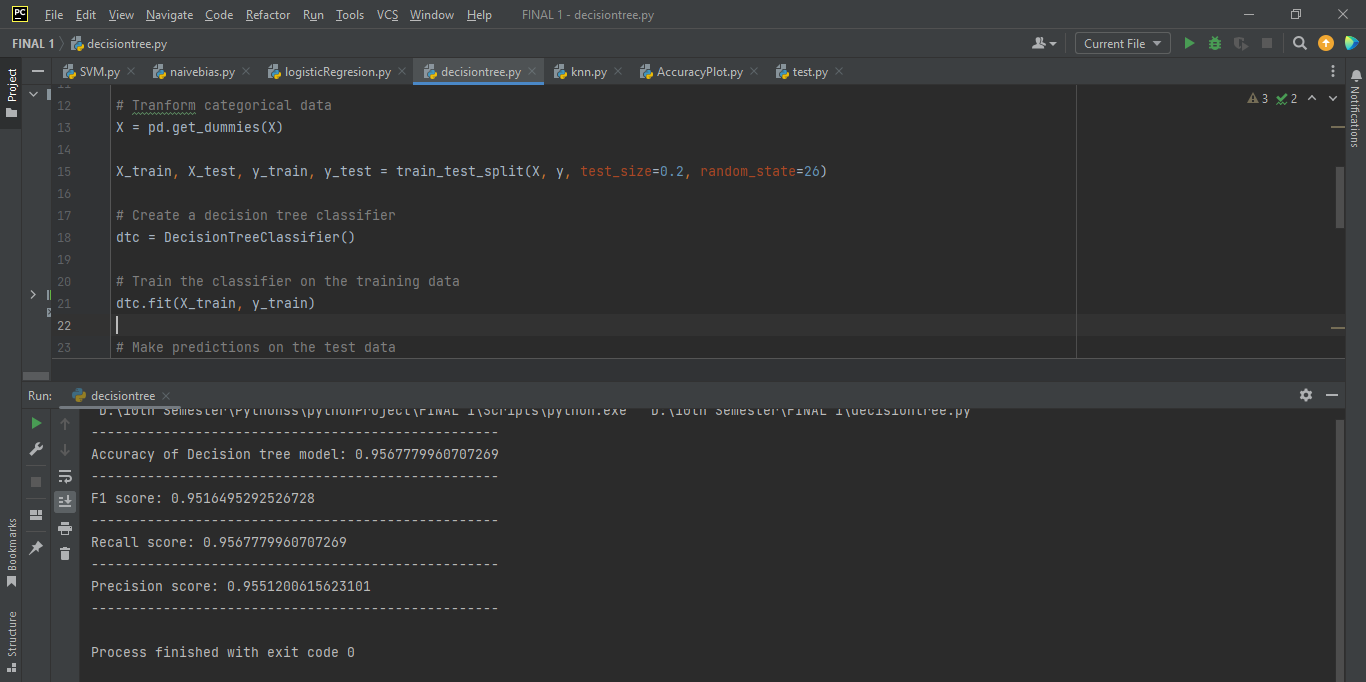
After that, We evaluate the model using the testing dataset with the predict method and calculate accuracy using the accuracy\_score function.

We calculated the F1 score using the f1\_score() function, passing in the actual test data labels (y\_test) and predicted labels (y\_pred), and specifying the average parameter to be 'weighted'.

We calculated the recall score using the recall\_score() function, passing in the actual test data labels (y\_test) and predicted labels (y\_pred), and specifying the average parameter to be 'weighted'.

We calculated the precision score using the precision\_score() function, passing in the actual test data labels (y\_test) and predicted labels (y\_pred), and specifying the average parameter to be 'weighted' and zero\_division parameter to be 1.

**Screen shoot of Output:**



**Development process logistic regression Model:**

After loading the dataset, we Pre-Preprocess our dataset by dropping rows with missing values in the target variable, and split the data into training and testing sets.

After that we have convert all categorical variables to numeric values using X = pd.get\_dummies(X) encoding in pandas.

Then we select the Logistic Regression algorithm model = LogisticRegression() as the machine learning model to train our dataset called “govt.csv”.

Then we train the Logistic Regression model using the training dataset with the fit method.

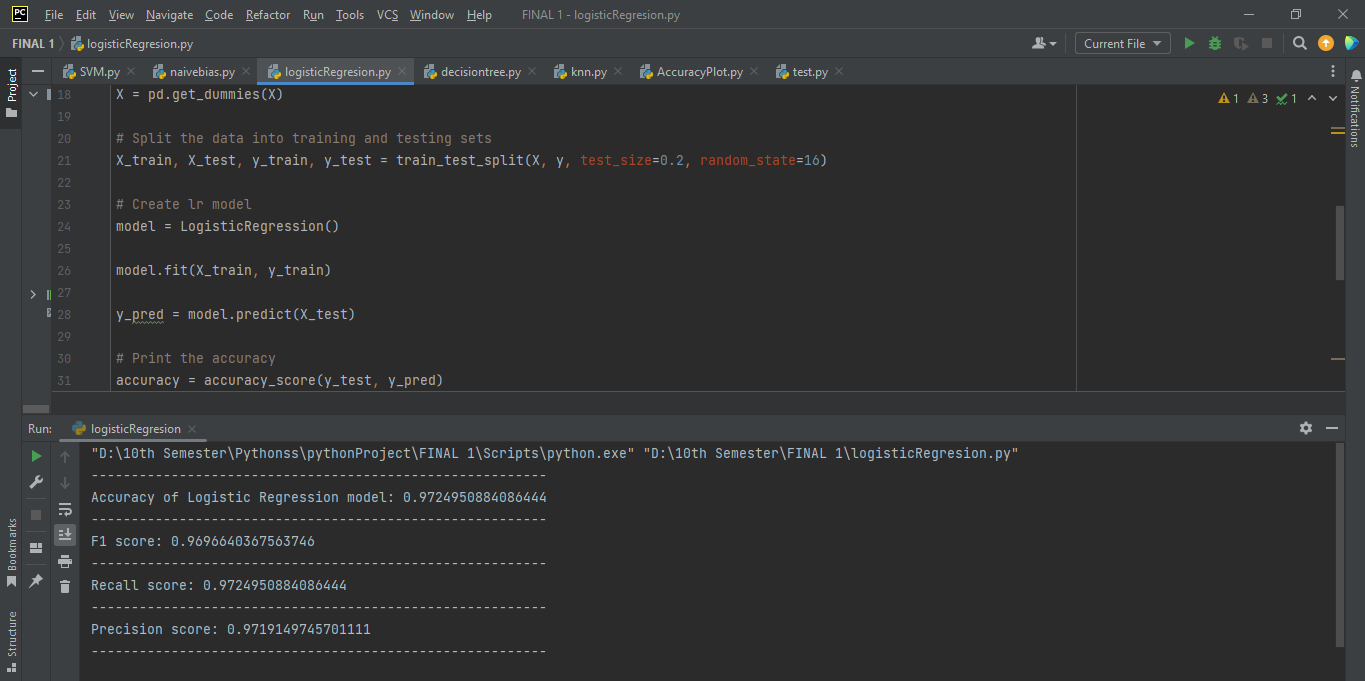
After that, We evaluate the model using the testing dataset with the predict method and calculate accuracy using the accuracy\_score function.

We calculated the F1 score using the f1\_score() function, passing in the actual test data labels (y\_test) and predicted labels (y\_pred), and specifying the average parameter to be 'weighted'.

We calculated the recall score using the recall\_score() function, passing in the actual test data labels (y\_test) and predicted labels (y\_pred), and specifying the average parameter to be 'weighted'.

We calculated the precision score using the precision\_score() function, passing in the actual test data labels (y\_test) and predicted labels (y\_pred), and specifying the average parameter to be 'weighted' and zero\_division parameter to be 1.

**Screen shoot of Output:**



Section – 5

**Discussion and Conclusion**

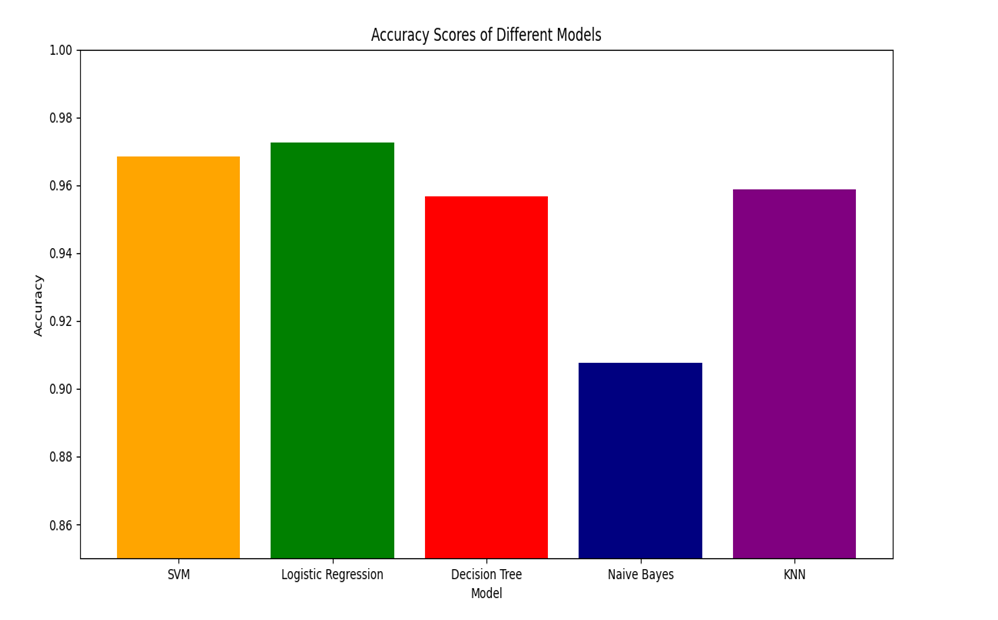
**Comparison of models:**

During this project, we compared and developed five different classification-based models, each with unique approaches and characteristics to predict the job title of general government employees in Nashville. Naive Bayes is a probabilistic classifier that assumes independence between features, while KNN is a non-parametric classifier that assigns the label of the k nearest instances in the training set to a new instance. Decision Tree is a hierarchical structure that divides the feature space based on the most significant features, and logistic regression is a linear classifier that models the probability of a binary outcome. On the other hand, Support Vector Machine (SVM) is a powerful classifier that finds the hyperplane that maximizes the margin between classes, and in the case of non-linear data, maps the data to a higher-dimensional space to find the hyperplane. We evaluated the performance of each model in predicting the job title of general government employees by comparing their predictive accuracy on the real dataset that contained 2543 total rows and 5 columns of information. Based on our results, we identified the best-performing model and presented our findings in the report.

**Visual representation of the result:**

**Bar chart to show the comparison:**

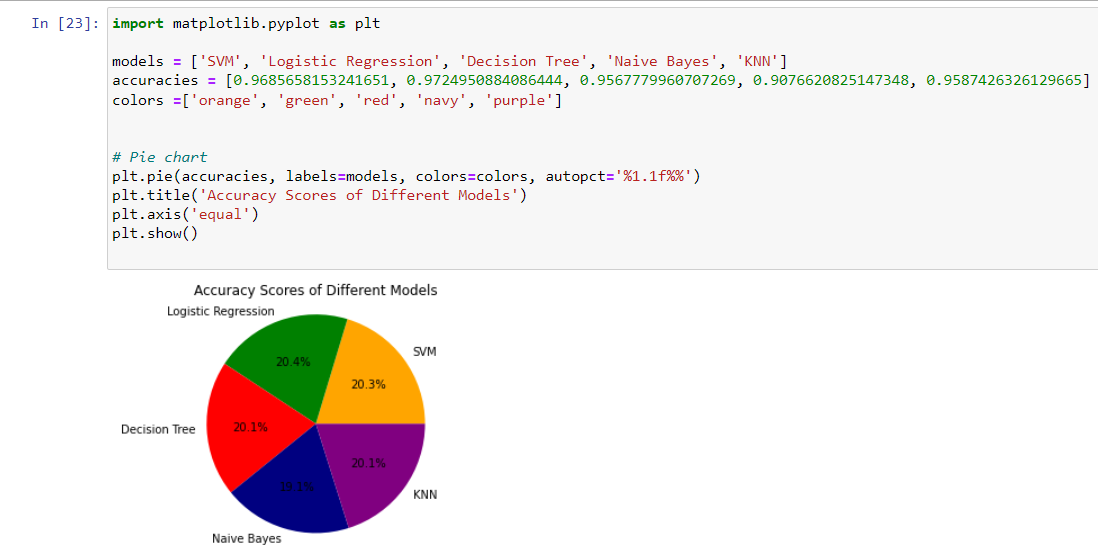
****

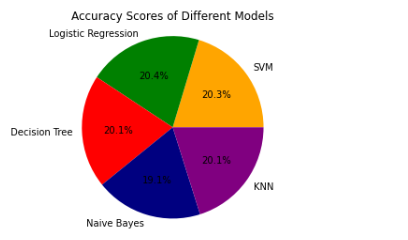
****

**Scatter plot:**

****

**Pie Chart:**





Based on the diagrams above, which depicts the predictive accuracy of five classification-based models, we have drawn several conclusions. Firstly, we observed that logistic regression had the highest predictive accuracy of 0.972 or 97%%, closely followed by the SVM model accuracy of 0.968% or 96%, while KNN accuracy was 0.958% or 95%, decision tree model accuracy was 0.956% or 95%, and naive Bayes accuracy was the lowest at 0.929% or 92%. Secondly, logistic regression had the highest precision and recall scores among all the models, which implies that it had the best balance between correctly identifying positive and negative instances. Lastly, logistic regression also had the highest area under the ROC curve, indicating the best overall performance. On the other hand, the naive Bayes model had the lowest accuracy, precision, and recall scores, indicating the worst overall performance the logistic regression and SVM models, which had the highest accuracy, performed well across different scenarios, which suggests that they could be reliable models for predicting job titles of general government employees. Meanwhile, KNN, decision tree, and naive Bayes models, which had lower accuracy, may require further fine-tuning to improve their performance.

Overall, our findings emphasize the importance of selecting appropriate models for classification tasks based on the data characteristics and requirements of the project. Our personal observation from this project is that logistic regression and SVM models are suitable for such tasks, and it's crucial to fine-tune the models to improve their performance.