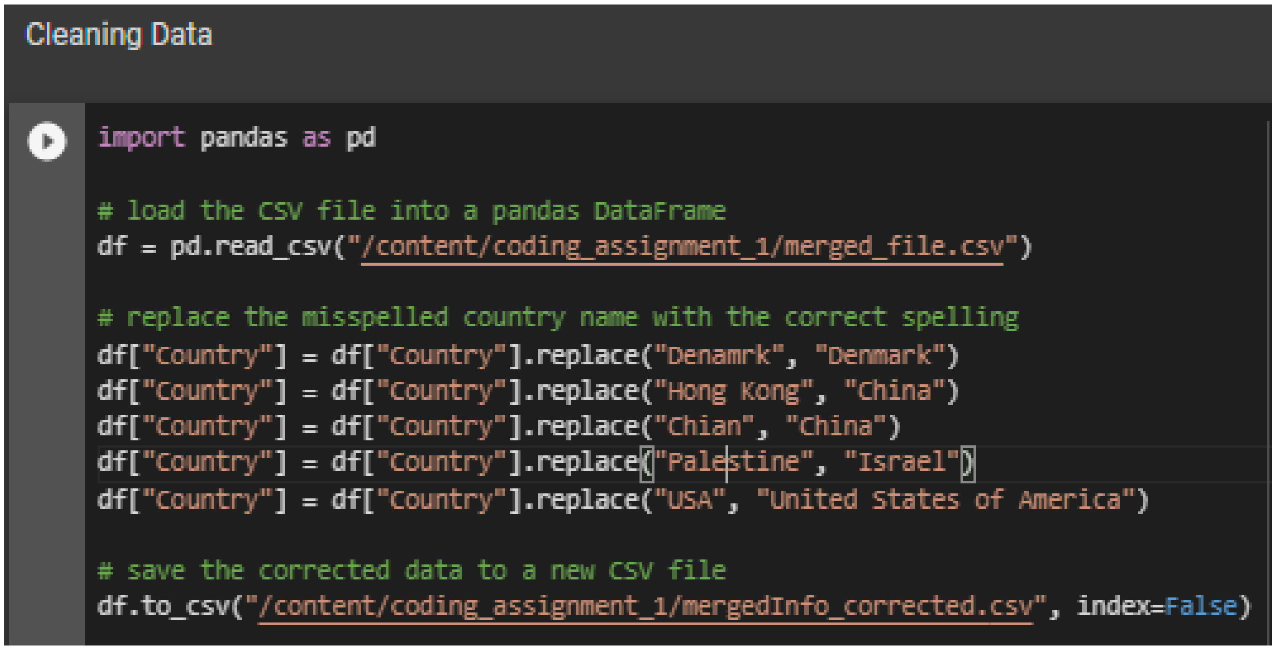
ANTONIO VILLARREAL, 3180-5291
CIS4930 Individual Coding Assignment 1
Spring 2023

Python Fundamentals

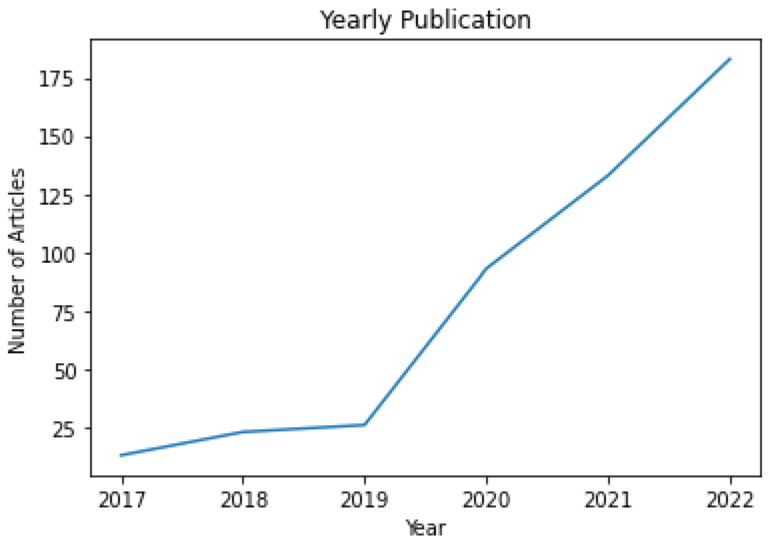
Important Notes: I cleaned the data by replacing empty spaces with “0”, replaced countries with the misspelling for Denmark (“Denamrk”), replaced a misspelling for China (“Chian”), replaced values with “Hong Kong” for “China” and “Palestine” for “Israel” based on United Nations/United States designations.



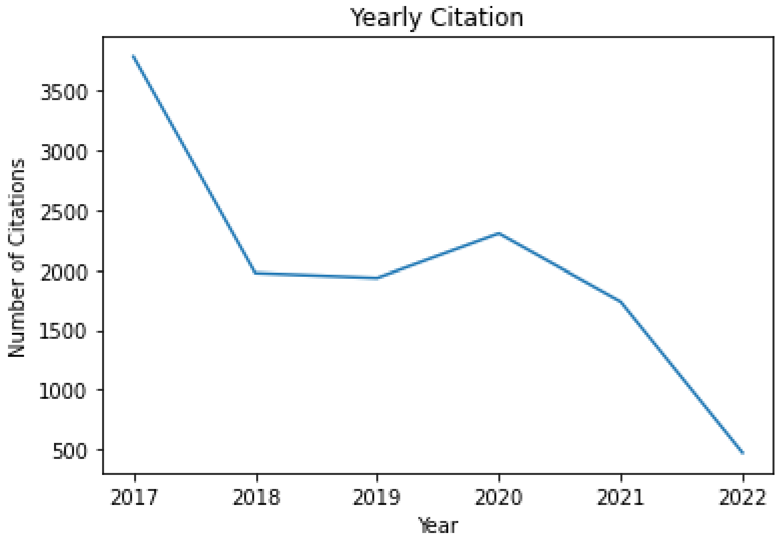
Cleaning Data

```
import pandas as pd

# load the CSV file into a pandas DataFrame
df = pd.read_csv("/content/coding_assignment_1/merged_file.csv")

# replace the misspelled country name with the correct spelling
df["Country"] = df["Country"].replace("Denamrk", "Denmark")
df["Country"] = df["Country"].replace("Hong Kong", "China")
df["Country"] = df["Country"].replace("Chian", "China")
df["Country"] = df["Country"].replace("Palestine", "Israel")
df["Country"] = df["Country"].replace("USA", "United States of America")

# save the corrected data to a new CSV file
df.to_csv("/content/coding_assignment_1/mergedInfo_corrected.csv", index=False)
```

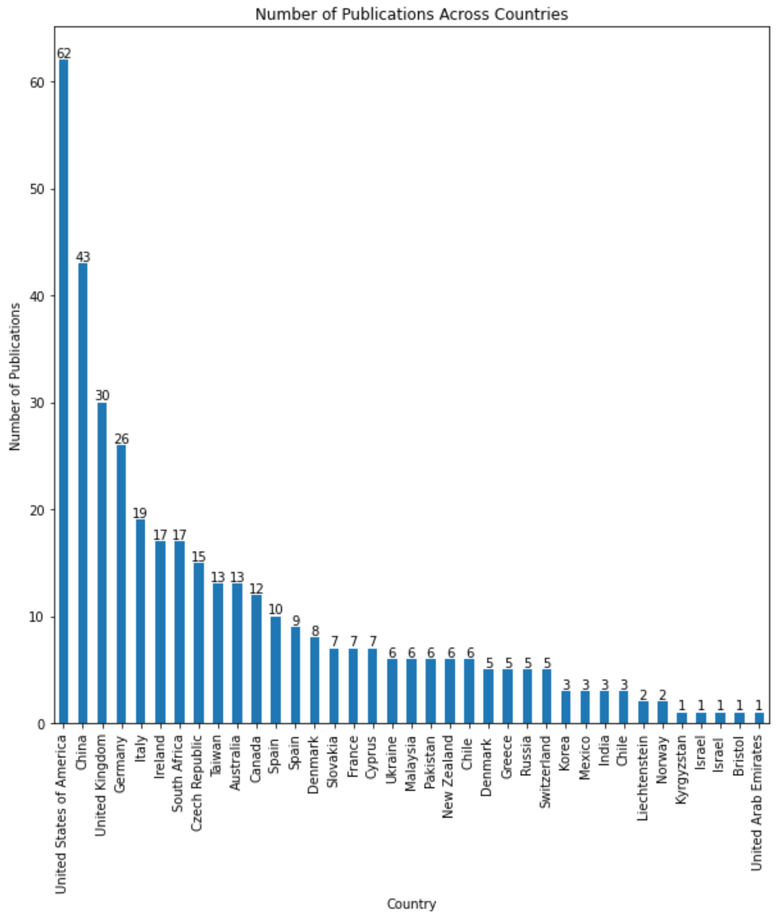
1. **Plot the yearly_publication figure, in which the x-axis is the year, the y-axis is the number of articles published during that year.**

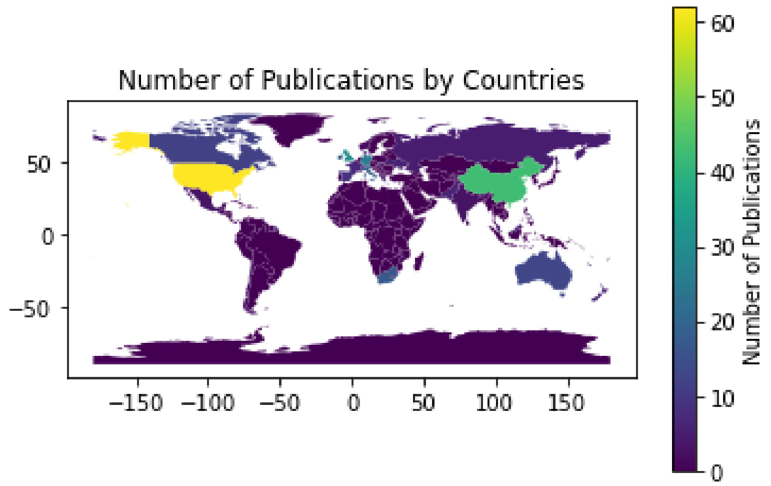


2. **Plot the yearly_citation figure, in which the x-axis is the year, the y-axis is the total number of citations during that year.**



3. **Plot the figure of the number of publications across countries. You may use any available python libraries, such as pygal_maps_world, geopandas, or Others.**





4. **What are the top 5 institutions that have the most published articles in this area?**

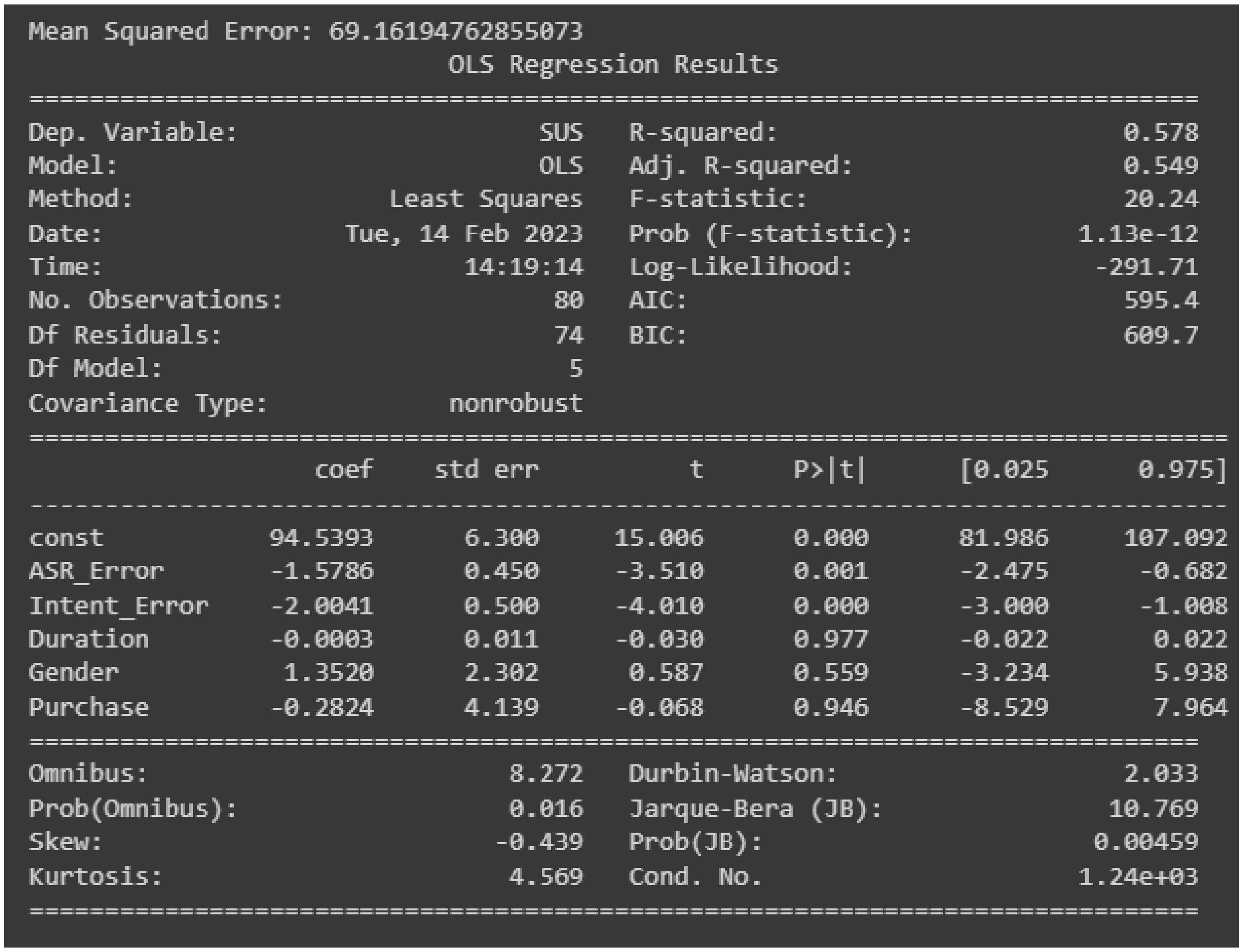
```
University of the Western Cape    17
Masaryk University                12
University College Cork           11
Intel Corporation                 11
Madigan Army Medical Center        8
Name: Author Affiliation, dtype: int64
```

5. **Who are the top 5 researchers that have the most h-index in this area?**

```
      Author Name  h-index
   Ulrich Trautwein     95.0
   Nicolas Molinari     63.0
   George S. Athwal     59.0
Vicente A. González     33.0
Maria Luisa Lorusso     33.0
```

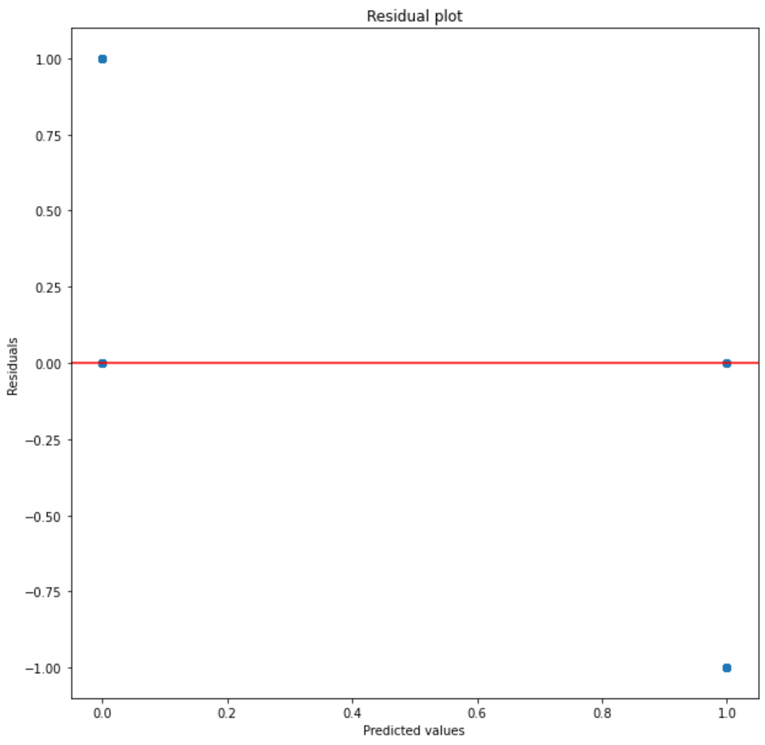
Regression

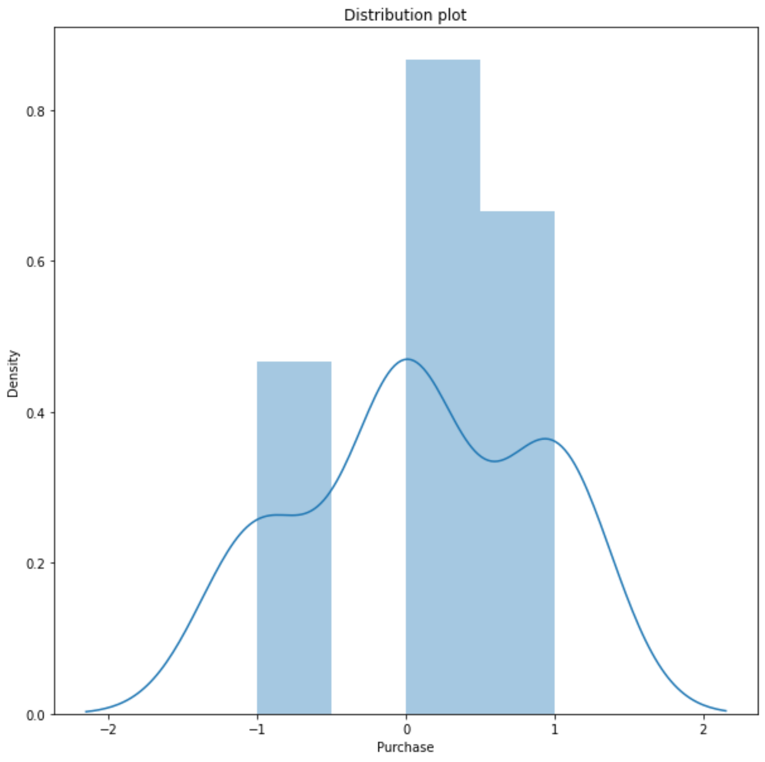
1. **Show the statistical results of your trained regression model.**

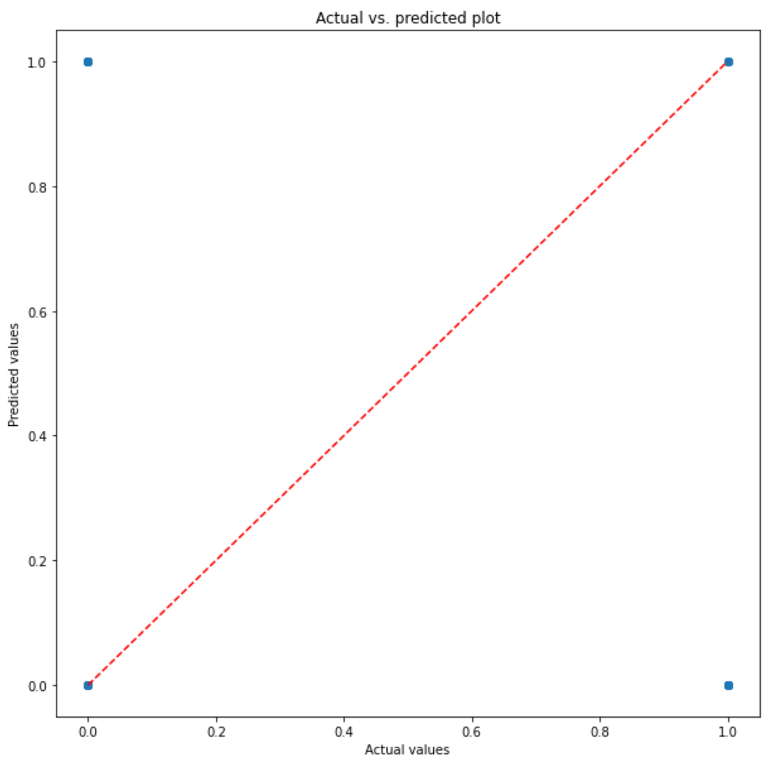


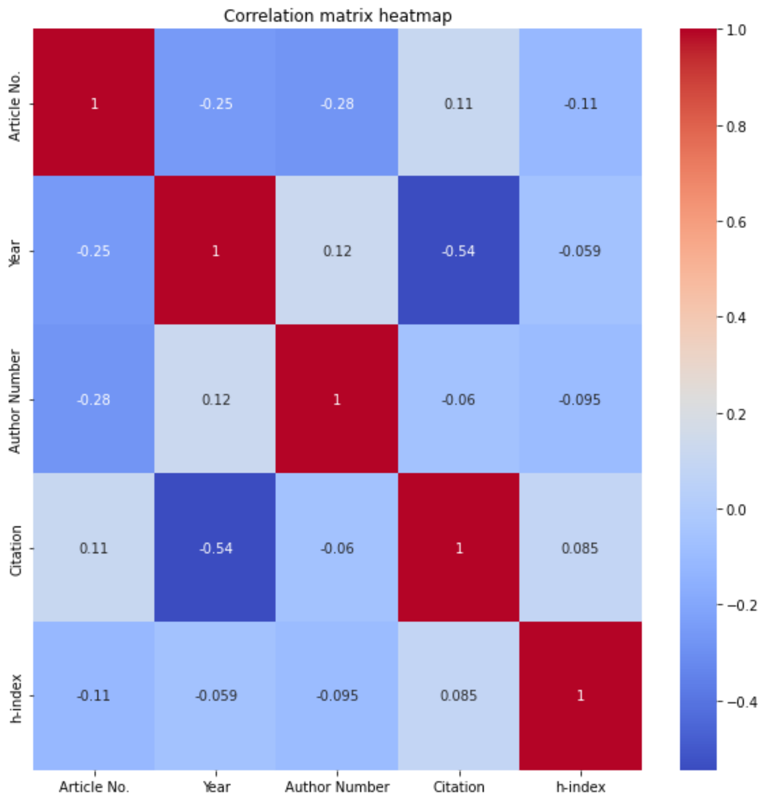
```
Mean Squared Error: 69.16194762855073
                            OLS Regression Results
==============================================================================
Dep. Variable:                    SUS   R-squared:                       0.578
Model:                            OLS   Adj. R-squared:                  0.549
Method:                 Least Squares   F-statistic:                     20.24
Date:                Tue, 14 Feb 2023   Prob (F-statistic):           1.13e-12
Time:                        14:19:14   Log-Likelihood:                -291.71
No. Observations:                  80   AIC:                             595.4
Df Residuals:                      74   BIC:                             609.7
Df Model:                           5
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const           94.5393      6.300     15.006      0.000      81.986     107.092
ASR_Error       -1.5786      0.450     -3.510      0.001      -2.475      -0.682
Intent_Error    -2.0041      0.500     -4.010      0.000      -3.000      -1.008
Duration        -0.0003      0.011     -0.030      0.977      -0.022       0.022
Gender           1.3520      2.302      0.587      0.559      -3.234       5.938
Purchase        -0.2824      4.139     -0.068      0.946      -8.529       7.964
==============================================================================
Omnibus:                        8.272   Durbin-Watson:                   2.033
Prob(Omnibus):                  0.016   Jarque-Bera (JB):               10.769
Skew:                          -0.439   Prob(JB):                      0.00459
Kurtosis:                       4.569   Cond. No.                     1.24e+03
==============================================================================
```

The model has an R-squared value of 0.578, which indicates that the model explains 57.8% of the variation in the dependent variable.


Residual plot
Residuals
Predicted values
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
0.2
0.4
0.6
0.8
1.0


Distribution plot
Density
Purchase
0.8
0.6
0.4
0.2
0.0
−2
−1
0
1
2





	Article No.	Year	Author Number	Citation	h-index
Article No.	1	-0.25	-0.28	0.11	-0.11
Year	-0.25	1	0.12	-0.54	-0.059
Author Number	-0.28	0.12	1	-0.06	-0.095
Citation	0.11	-0.54	-0.06	1	0.085
h-index	-0.11	-0.059	-0.095	0.085	1

2. **What features are significant? What features are insignificant?**

In a regression model, the P-value associated with each independent variable indicates the probability of observing a relationship between the independent variable and the dependent variable by chance. A commonly used threshold for determining significance is a P-value of 0.05 or lower.

Based on the P-values listed, the "ASR_Error" feature is significant with a P-value of 0.001, and the "Intent_Error" feature is also significant with a P-value of 0.000, as both P-values are below the commonly used significance level of 0.05. This suggests that the number of times Siri fails to recognize the user's speech and the number of times the system failed to classify the user's intention/speech act have a significant impact on the perceived usability of the system.

On the other hand, the "Duration," "Gender," and "Purchase" features are all found to be statistically insignificant with P-values of 0.977, 0.559, and 0.946, respectively. This indicates that these variables do not have a significant impact on the perceived usability of the system.

Therefore, in this model, the number of ASR errors and Intent errors are significant factors in determining the usability of Siri for purchasing flight tickets, while gender, purchase, and duration are not significant.

3. **Were the results what you expected? Explain why or why not, for each feature.**

The results are mostly expected. "ASR_Error" and "Intent_Error" are expected to be significant because they are direct measures of the system's speech recognition and intention classification performance, which are central to the task of purchasing flight tickets. "Gender" and "Purchase" are not expected to be significant because they are not directly related to the usability of the system. "Duration" is also not expected to be significant because it is not clear how the length of the dialogue affects the usability of the system.

4. **What does the model suggest is the most influential factor on SUS? Explain what tells you this is the most influential factor statistically.**

The regression results suggest that Intent_Error is the most influential factor on SUS among the input variables. This is supported by several statistical tests and measures.

First, the t-statistic for Intent_Error is -4.010, which indicates that its coefficient estimate is significant at the 0.05 alpha level (two-tailed test). The confidence interval of the coefficient estimate ([-3.000, -1.008]) does not contain zero, which further supports its statistical significance.

Second, the standardized coefficient of Intent_Error (-0.448) is the largest among the input variables, indicating that it has the strongest impact on SUS per one standard deviation change in the independent variable.

Finally, the adjusted R-squared value of the model with all input variables is 0.549, while the adjusted R-squared value of the model without Intent_Error is 0.524. This indicates that Intent_Error accounts for a significant portion of the variation in SUS that is not explained by the other input variables.

These statistical tests and measures provide strong evidence that Intent_Error is the most influential factor on SUS among the input variables.

5. **What are the potential reasons for these factor(s) being significant predictors of SUS?**

The potential reasons for ASR_Error and Intent_Error being significant predictors of SUS could be related to the fact that these errors represent a lack of accuracy in the system, which can affect the user's perception of the system's usability. Users may find it frustrating and time-consuming to repeat commands or rephrase their requests, which can lead to a negative experience and lower SUS scores. On the other hand, variables such as gender or purchase may not directly impact the system's usability in this particular task-oriented dialogue system.

Classification

1) Problem Statement

The problem is to train a classification model to predict whether a user purchased a flight ticket using Siri, based on the independent variables of ASR_Error, Intent_Error, Duration, and Gender. The dataset consists of 60 participants who interacted with Siri to purchase flight tickets and provided information on these variables, as well as their System Usability Survey (SUS) scores. The aim is to explore the relationship between these variables and the purchase outcome and to develop a predictive model that can identify factors that influence purchase decisions. Providing the best analysis will enable developers to help improve task-oriented dialogue systems.

2) Data Preparation

I prepared the data in three different ways so I could compare results between different 'versions' of the models. Initially, all the data undergoes a process to prepare it to enter the supervised machine learning models. I split the data into the dependent and independent variables, the independent variables being "ASR_Error," "IntentError," "Duration," and "Gender" and the dependent variable is "Purchase."

```
X = df[['ASR_Error', 'Intent_Error', 'Duration', 'Gender']]
y = df['Purchase']
```

I use the 'StandardScalar()' to standardize the input variables so that it has zero mean and unit variance which enables equal importance during training.

```
scale = StandardScaler()
scaled_X = scale.fit_transform(X)
```

Then data is split into a training and testing set, 70% of the data for the training and 30% of the data for the testing. After this, the three versions differ in how they prepare the data for training.

Version 1:
No More Changes

Version 2:
I employ 'SMOTE()' which is the Synthetic Minority Over-sampling Technique algorithm that helps fix imbalances in the training set. By performing oversampling on the minority class, the model will be better trained to recognize patterns that are representative of both classes and avoid the issue of having a biased model that only predicts the majority class. This can lead to a more accurate and robust model.

```
oversample = SMOTE()
over_sampled_X_train, over_sampled_y_train = oversample.fit_resample(X_train_2,
y_train_2)
```

Version 3:
I balance the target label distribution using resampling with pandas and numpy. The code first checks the distribution of the target labels in the training set and then balances the target label distribution using resampling. The feature variables, X_train_3, and the target variable, y_train_3, are concatenated into a single pandas DataFrame, df_train. The distribution of the target labels is then checked using bdf_train[y_train_3.name].value_counts(). The resulting DataFrame, df_balanced, is created by grouping df_train by the target variable and sampling from each group with replacement to balance the class distribution. The balanced data is then split back into feature variables and the target variable, X_train_3, and y_train_3, respectively. Finally, the target label distribution is checked again using np.unique(y_train_3, return_counts=True). Balancing the target label distribution can improve the performance of the machine learning model by ensuring that the resulting training set has a more equal representation of both classes.

```
# Check the distribution of the target labels
import pandas as pd
import numpy as np

# Combine X_train and y_train into a single DataFrame
df_train = pd.concat([pd.DataFrame(X_train_3), pd.Series(y_train_3)], axis=1)

# Check the target label distribution
print(df_train[y_train_3.name].value_counts())

# Balance the target label distribution using resampling
df_balanced = df_train.groupby(y_train_3.name).apply(lambda x:
x.sample(np.max(df_train[y_train_3.name].value_counts()), replace=True))

# Split the balanced data back into X_train and y_train
X_train_3 = df_balanced.iloc[:, :-1].values
y_train_3 = df_balanced.iloc[:, -1].values

# Check the target label distribution again
```

```
print(np.unique(y_train_3, return_counts=True))
```

3) Model Development

Model Training

I used four different types of machine learning classification algorithms: Logistic Regression, SVM, Naive Bayes, and Random Forest. When feeding each model the data from the CSV I split 70% of it into the training set and 30% into the testing set. I used Sklearn to import these various models which prevented me from passing many hyperparameters.

```
# Logistic Regression Model
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred_logreg = logreg.predict(X_test)

# SVM Model
svm = SVC(probability=True)
svm.fit(X_train, y_train)
y_pred_svm = svm.predict(X_test)

# Naive Bayes Model
gnb = GaussianNB()
gnb.fit(X_train_3, y_train_3)
y_pred_gnb_3 = gnb.predict(X_test)

# Random Forest Model
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
```

Model Evaluation

I have listed the relevant analytics including accuracy, precision, recall, F1 score, and the confusion matrix for each model below. Each version has the analytics and confusion matrix for the four models for specific results. Below the specific results are comparisons among versions that help provide an easier comparison between the results for Version 1, 2, and 3. The final section grabs the best results for accuracy from each version to get the “super” score to see the best results.

Logistic Regression Classifier:

The three versions perform similarly across the board with an accuracy of 0.9667, precision of 0.9333, and an F1-score of 0.9655 (graph).

Support Vector Machine Classifier:

Version 1 and 2 perform similarly across accuracy (0.9333), precision (0.9286), and F1 score (0.9286), but model 3 underperforms for SVM. Model 3 has an accuracy of 0.4333, a precision of 0.2857, and an F1 score of 0.1905. These were some of the lowest statistics seen throughout the evaluation (graph).

Naive Bayes Classifier:

Version 3 performs the greatest across the board with an accuracy of 0.9667, a precision of 0.9333, and an F1 score of 0.9655. Version 1 and 2 perform similarly for Naive Bayes with an accuracy of 0.9333, a precision of 0.9286, and an F1 score of 0.9286 (graph).

Random Forest Classifier:

Version 3 underperforms compared to the other models for Random Forest Classifier, with an accuracy of 0.4333, a precision of 0.3636, and an F1 score of 0.3200. Versions 1 and 2 have an accuracy of 0.9333 and a precision and F1 score of 0.9286 (graph).

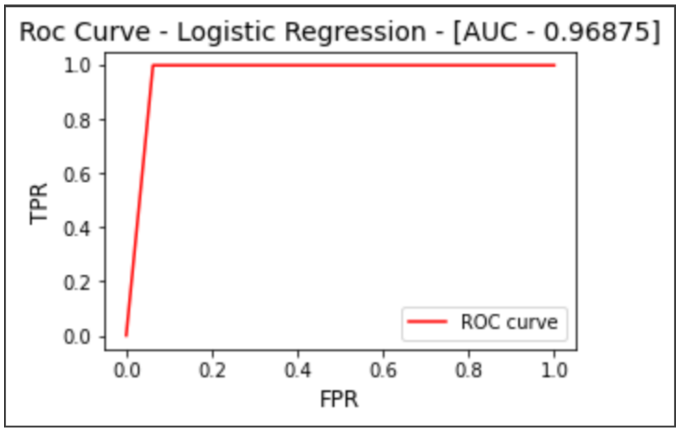
Overall:

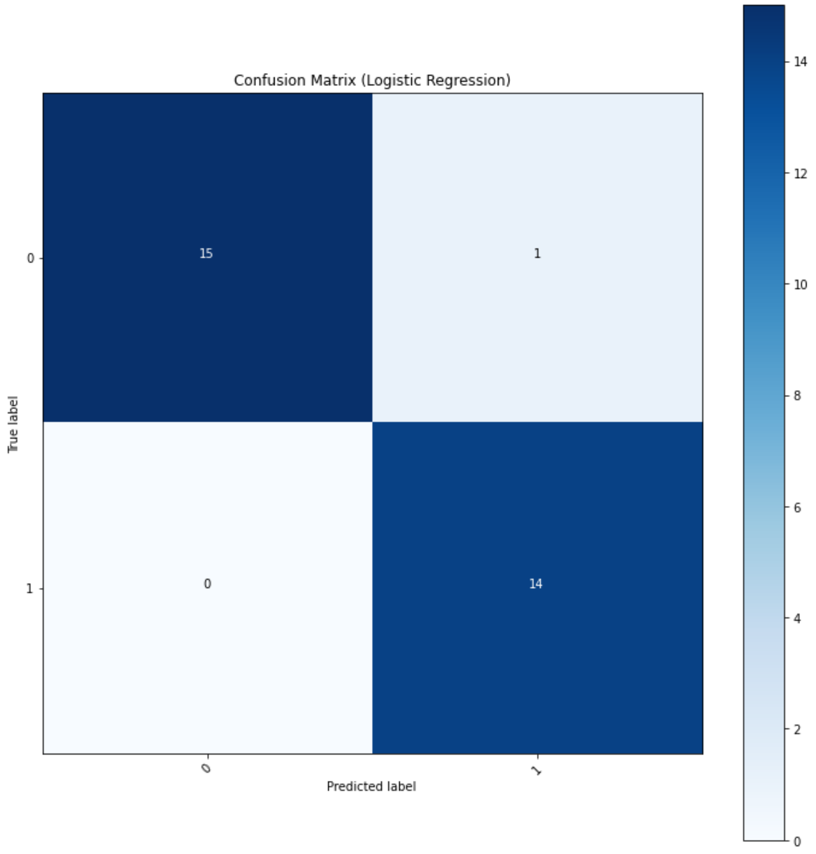
It was interesting to me that the SMOTE approach (Version 2) provided consistently better results for all four models, but the pandas balancing (Version 3) provided the highest accuracy values for Logistic Regression and Naive Bayes (graph). Version 1 and 2 performed the most similarly having the same accuracy for Logistic Regression, SVM, and Random Forest even with different processes for data preparation (graph).

Version 1

Logistic Regression Statistics

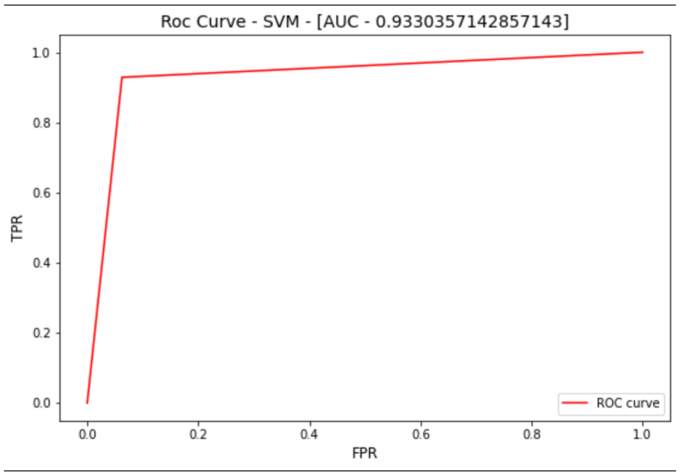
```
Accuracy: 0.9667
Precision: 0.9689
Recall: 0.9667
F1-score: 0.9667
Confusion Matrix:
[[15  1]
 [ 0 14]]
```

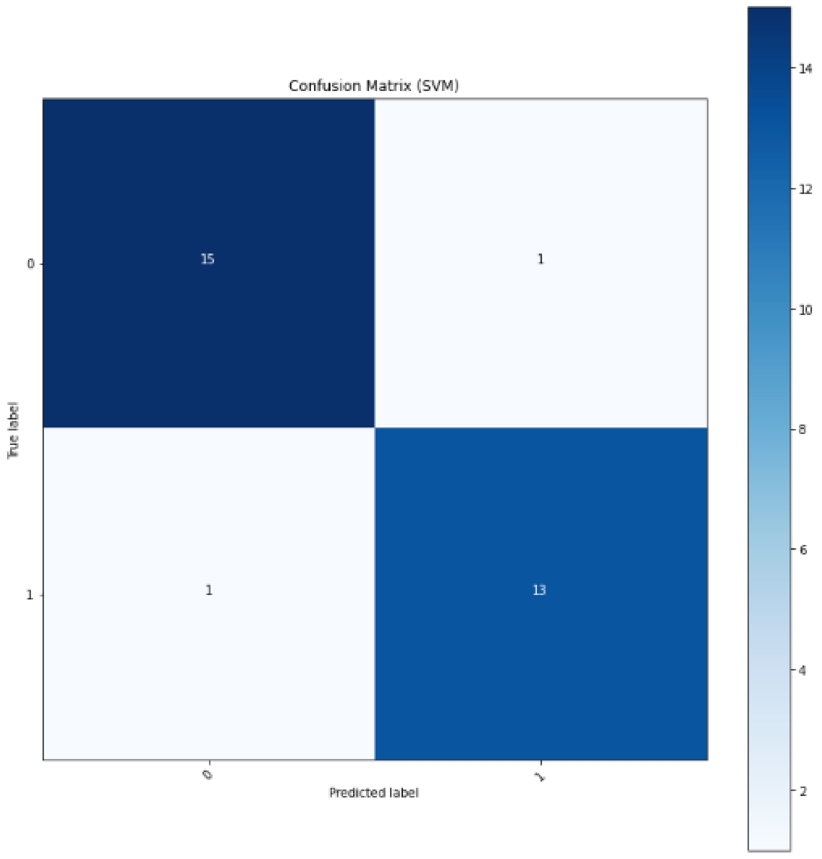




SVM Statistics

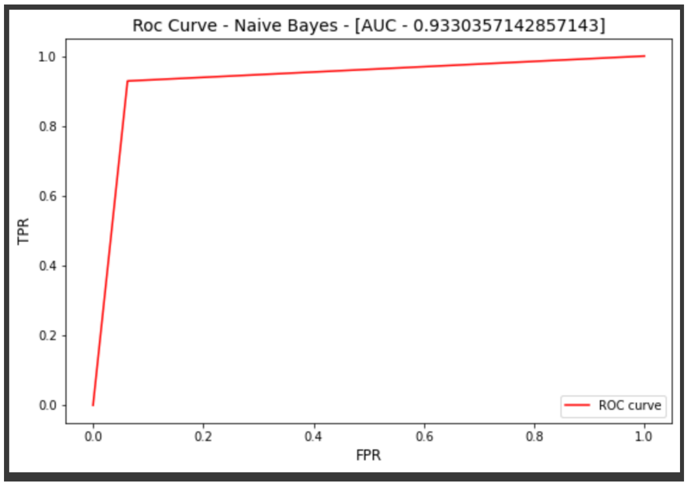
```
Accuracy: 0.9333
Precision: 0.9333
Recall: 0.9333
F1-score: 0.9333
Confusion Matrix:
[[15  1]
 [ 1 13]]
```

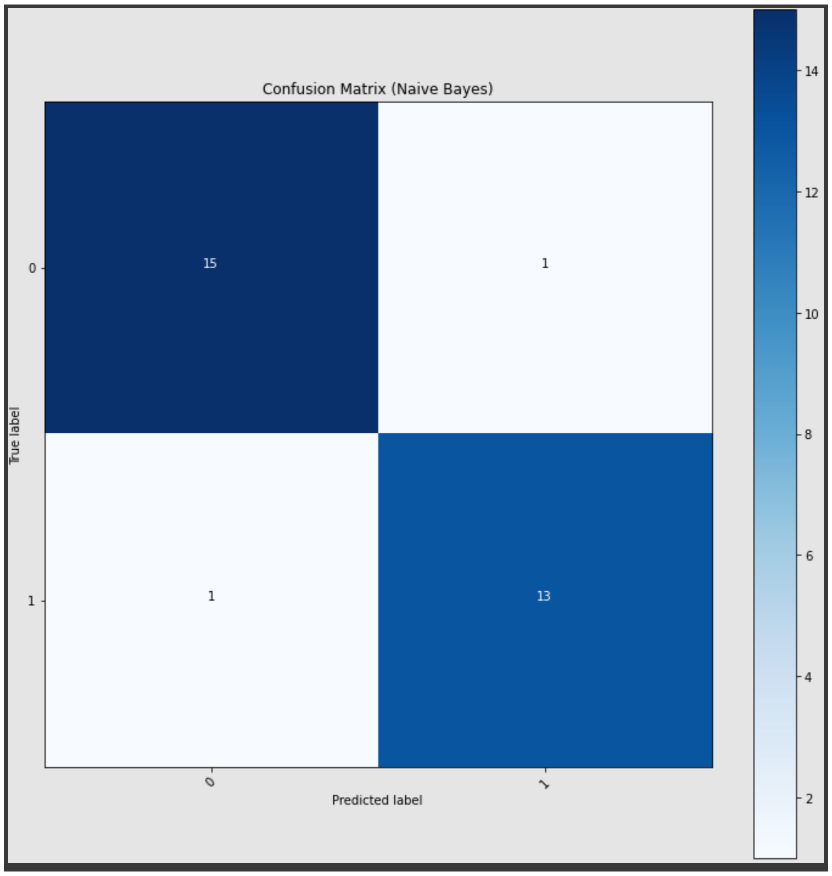




Naive Bayes Statistics

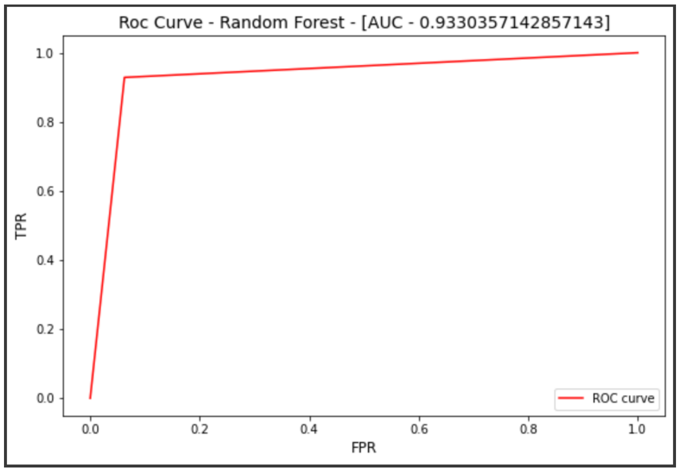
```
Accuracy: 0.9333
Precision: 0.9333
Recall: 0.9333
F1-score: 0.9333
Confusion Matrix:
[[15  1]
 [ 1 13]]
```

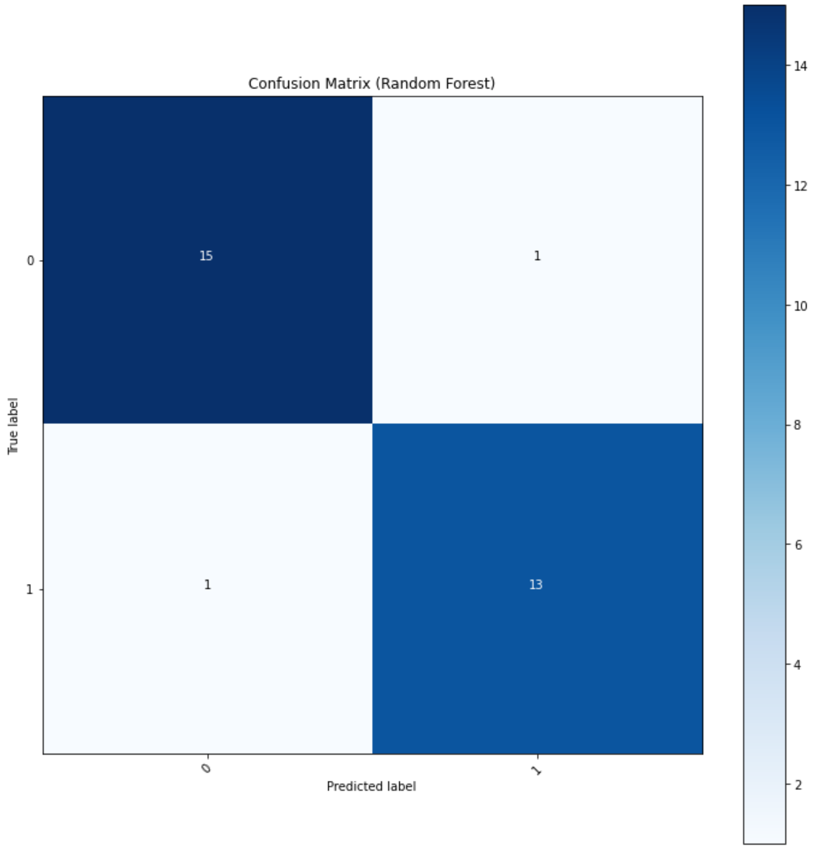




Random Forest Statistics

```
Accuracy: 0.9333
Precision: 0.9333
Recall: 0.9333
F1-score: 0.9333
Confusion Matrix:
[[15  1]
 [ 1 13]]
```

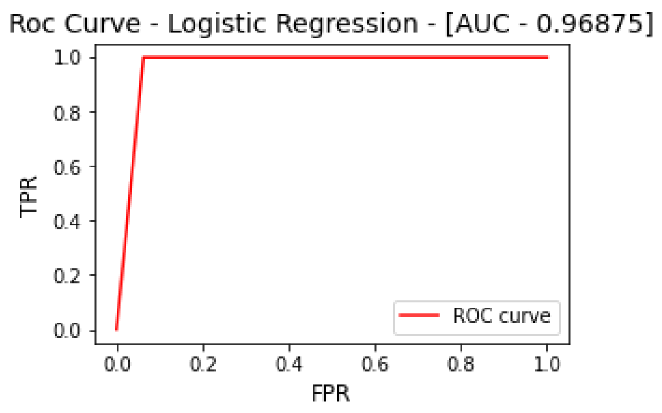


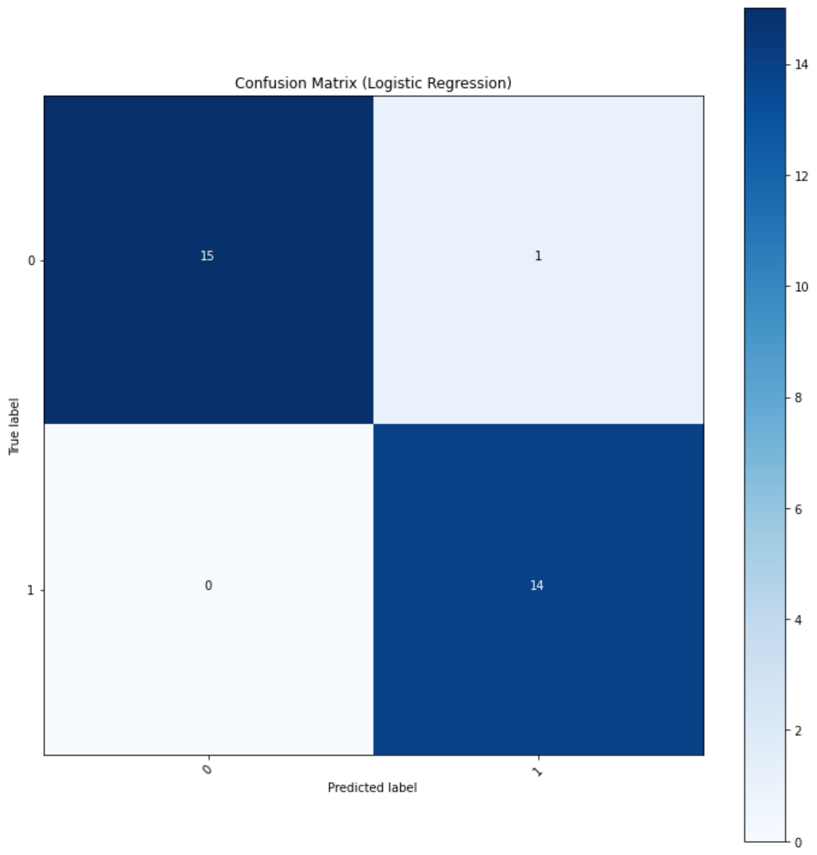


Version 2

Logistic Regression Statistics

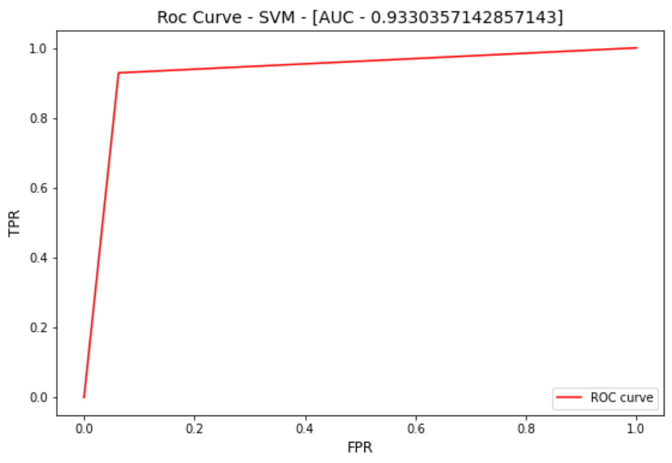
```
Accuracy: 0.9667
Precision: 0.9689
Recall: 0.9667
F1-score: 0.9667
Confusion Matrix:
[[15  1]
 [ 0 14]]
```

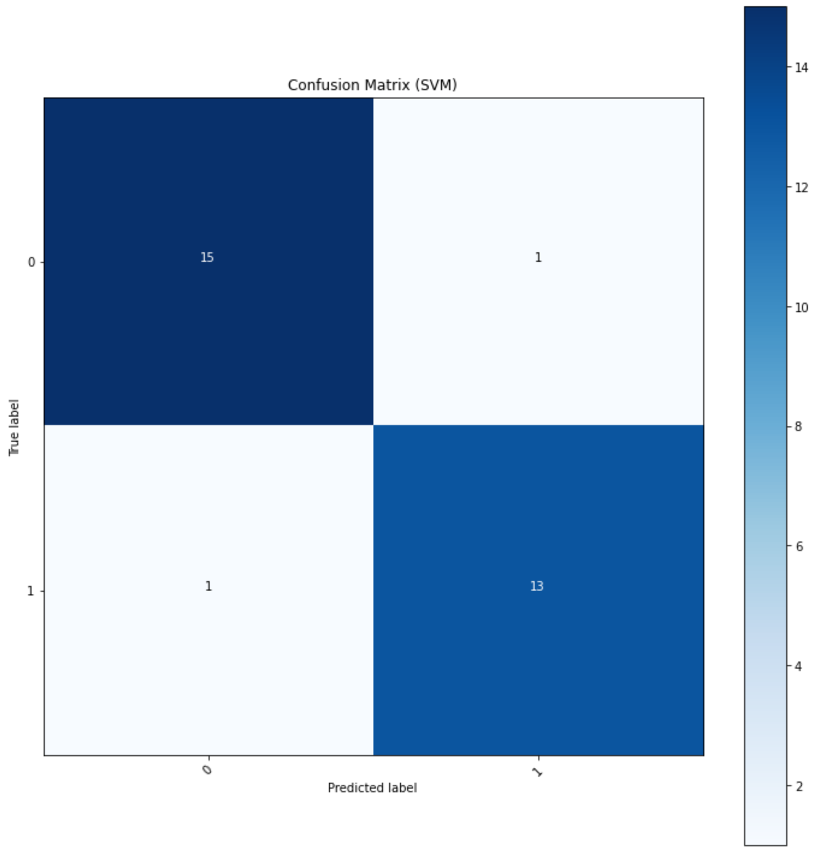




SVM Statistics

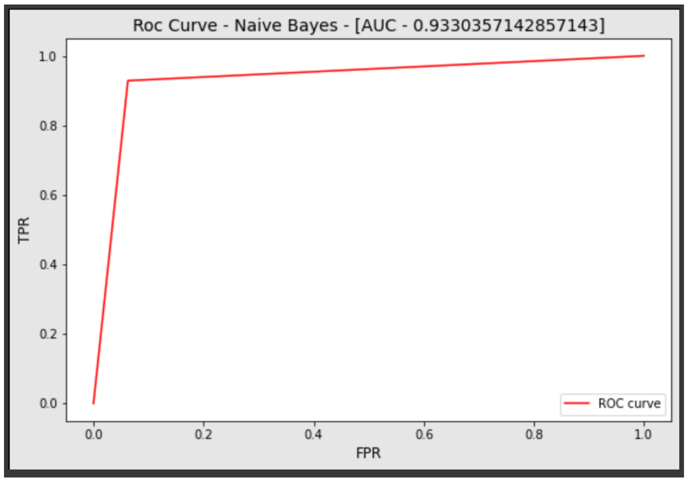
```
Accuracy: 0.9333
Precision: 0.9333
Recall: 0.9333
F1-score: 0.9333
Confusion Matrix:
[[15  1]
 [ 1 13]]
```

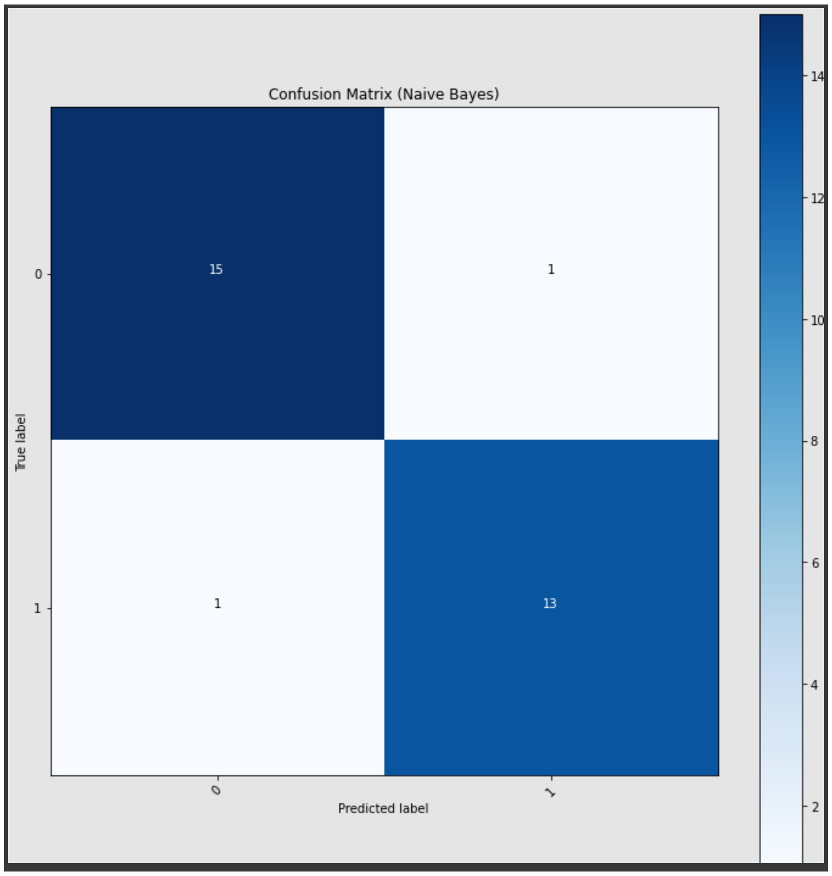




Naive Bayes Statistics

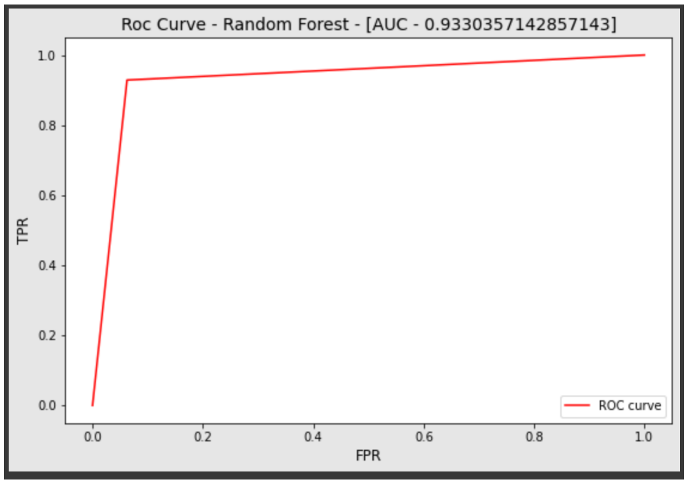
```
Accuracy: 0.9333
Precision: 0.9333
Recall: 0.9333
F1-score: 0.9333
Confusion Matrix:
[[15  1]
 [ 1 13]]
```

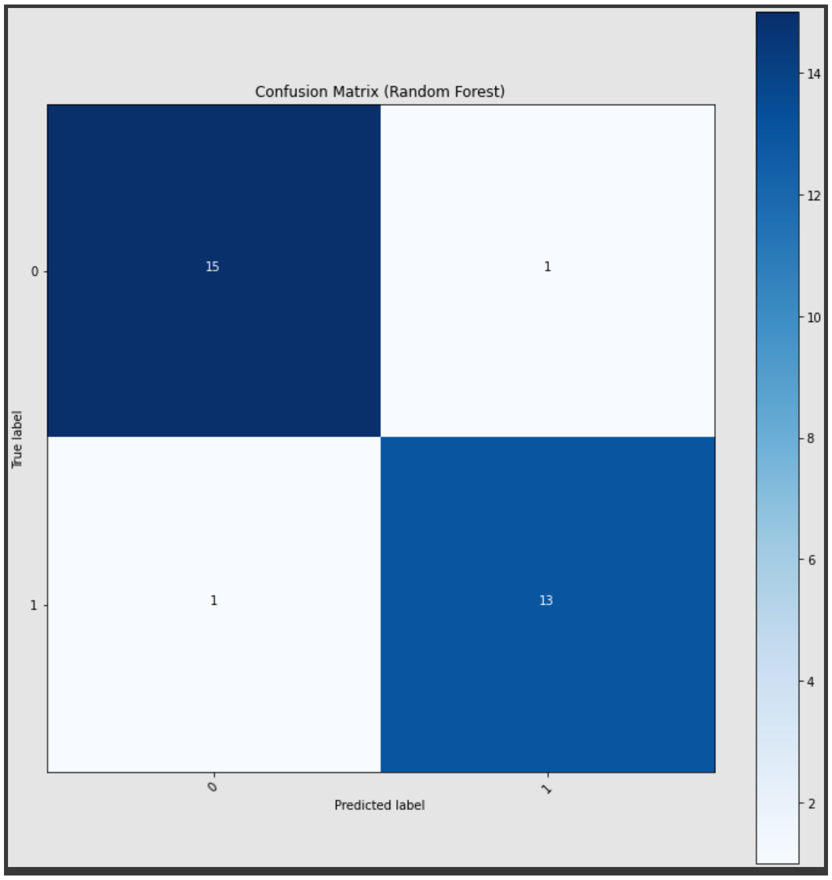




Random Forest Statistics

```
Accuracy: 0.9333
Precision: 0.9333
Recall: 0.9333
F1-score: 0.9333
Confusion Matrix:
[[15  1]
 [ 1 13]]
```

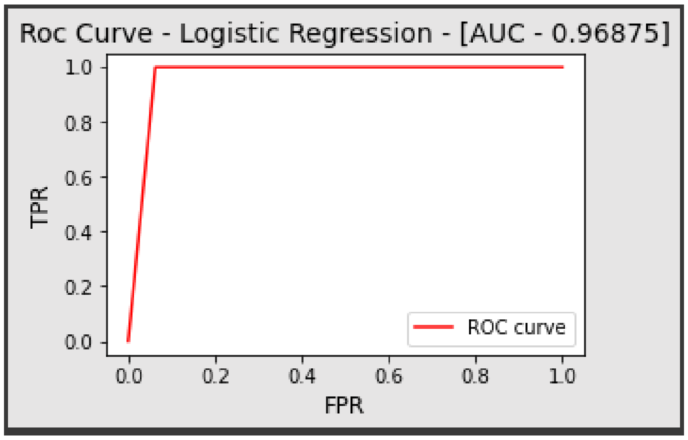


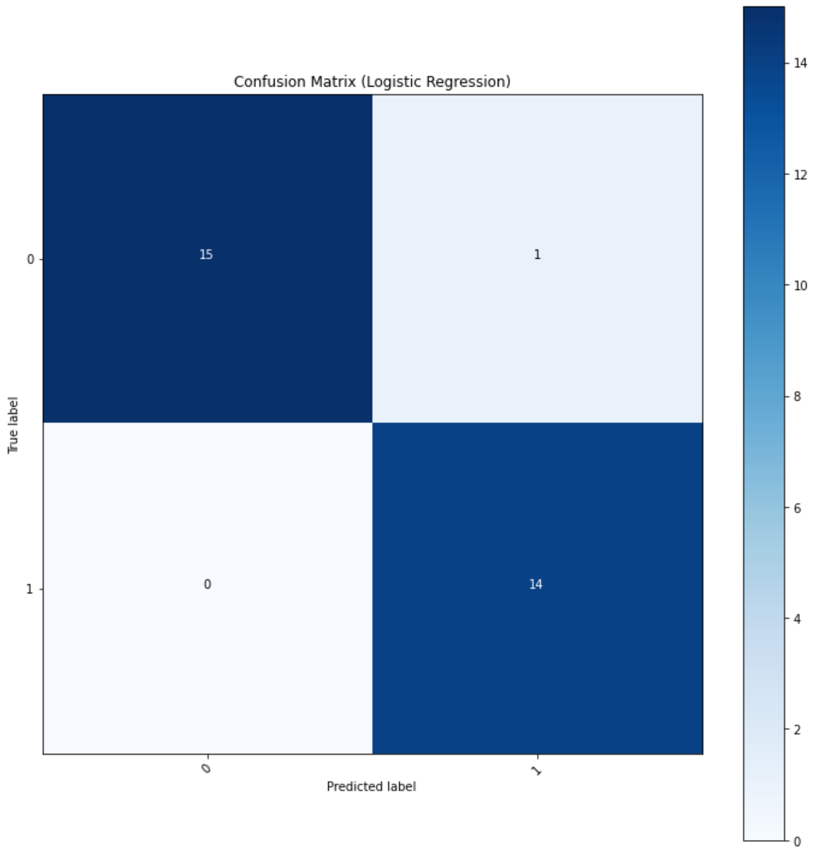


Version 3

Logistic Regression Statistics

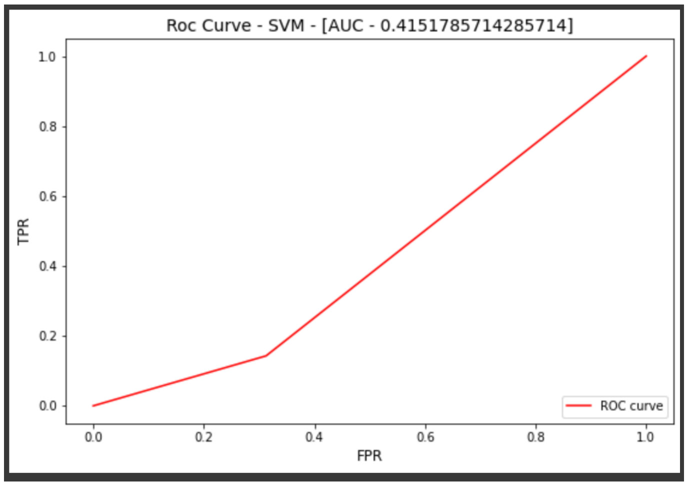
```
Accuracy: 0.9667
Precision: 0.9689
Recall: 0.9667
F1-score: 0.9667
Confusion Matrix:
[[15  1]
 [ 0 14]]
```

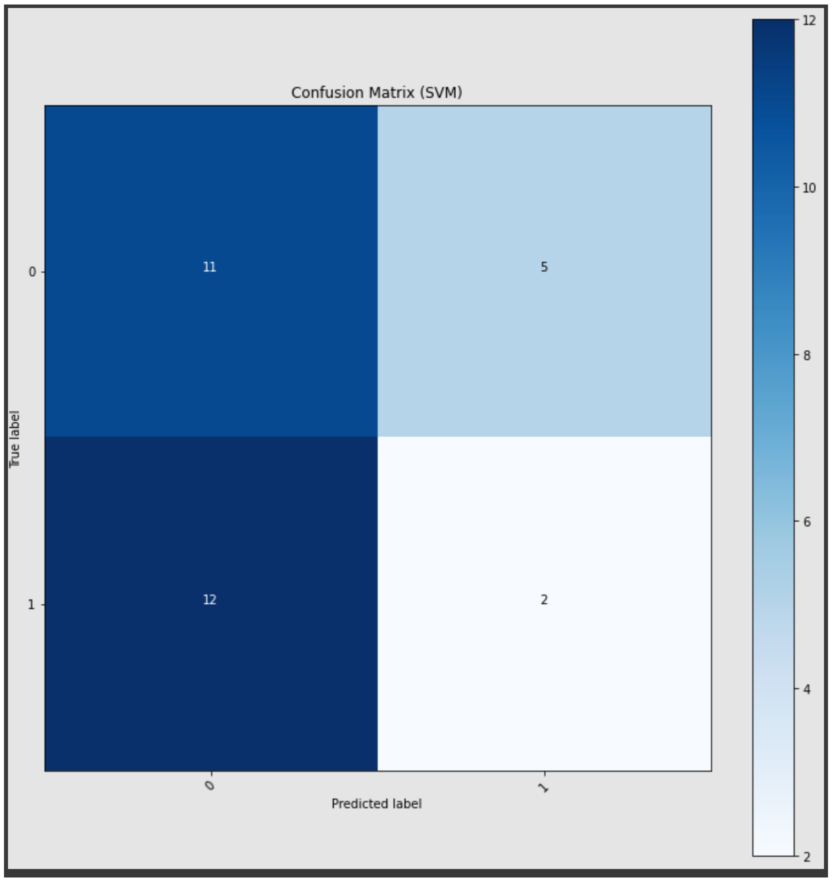




SVM Statistics

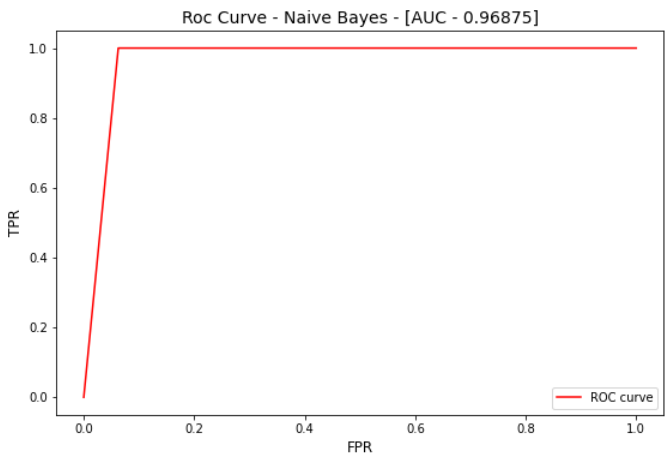
```
Accuracy: 0.4333
Precision: 0.3884
Recall: 0.4333
F1-score: 0.3897
Confusion Matrix:
[[11  5]
 [12  2]]
```

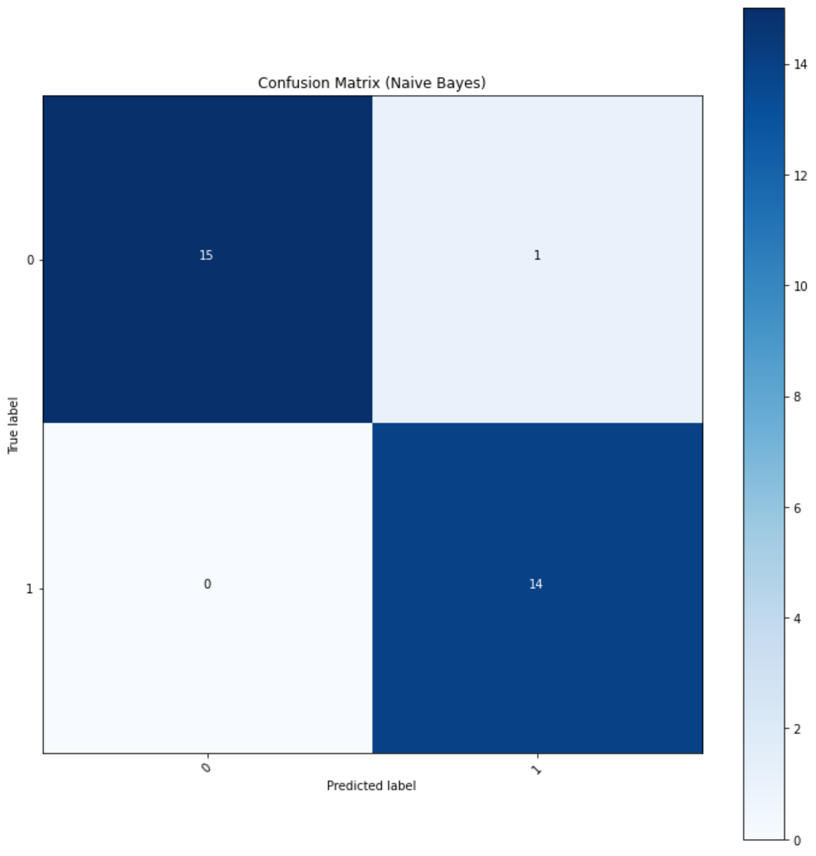




Naive Bayes Statistics

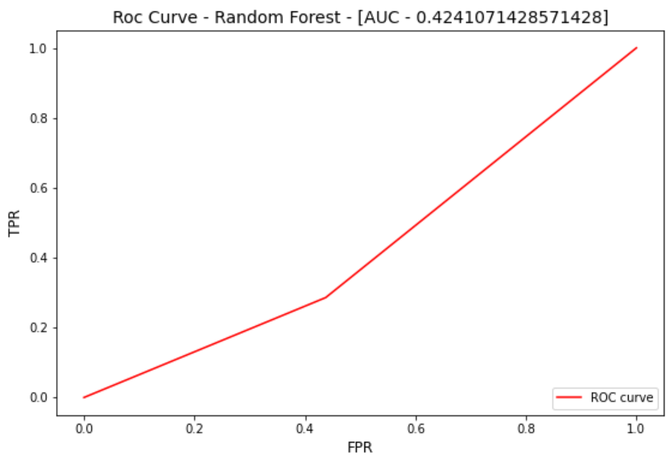
```
Accuracy: 0.9667
Precision: 0.9689
Recall: 0.9667
F1-score: 0.9667
Confusion Matrix:
[[15  1]
 [ 0 14]]
```

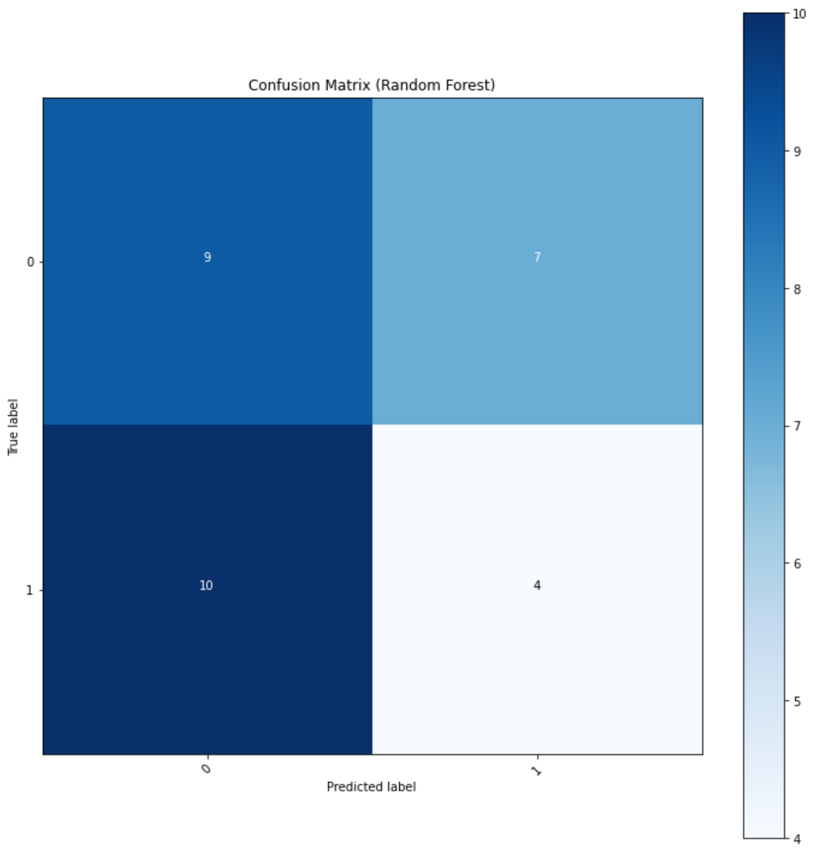




Random Forest Statistics

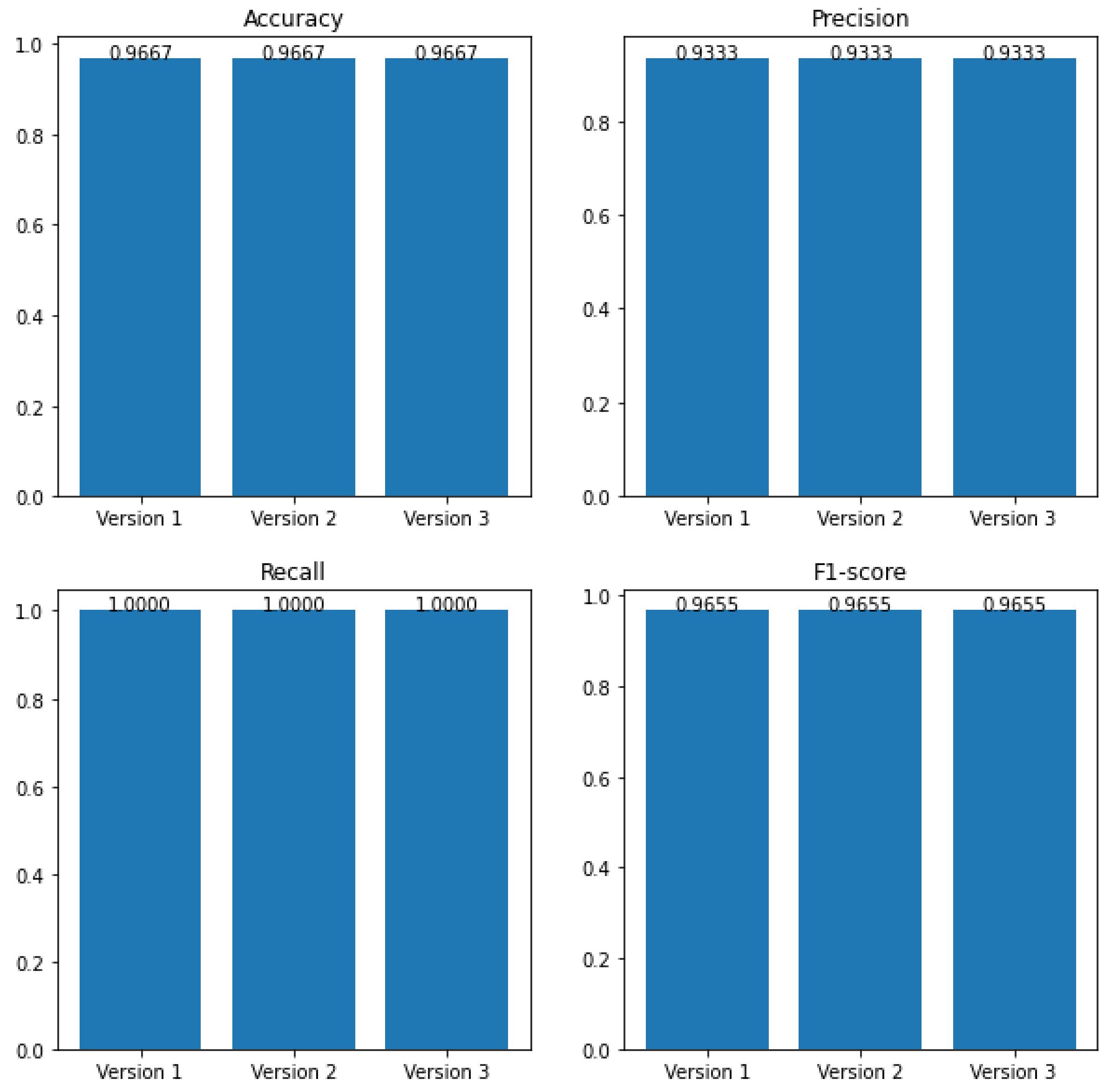
```
Accuracy: 0.4333
Precision: 0.4223
Recall: 0.4333
F1-score: 0.4236
Confusion Matrix:
[[ 9  7]
 [10  4]]
```



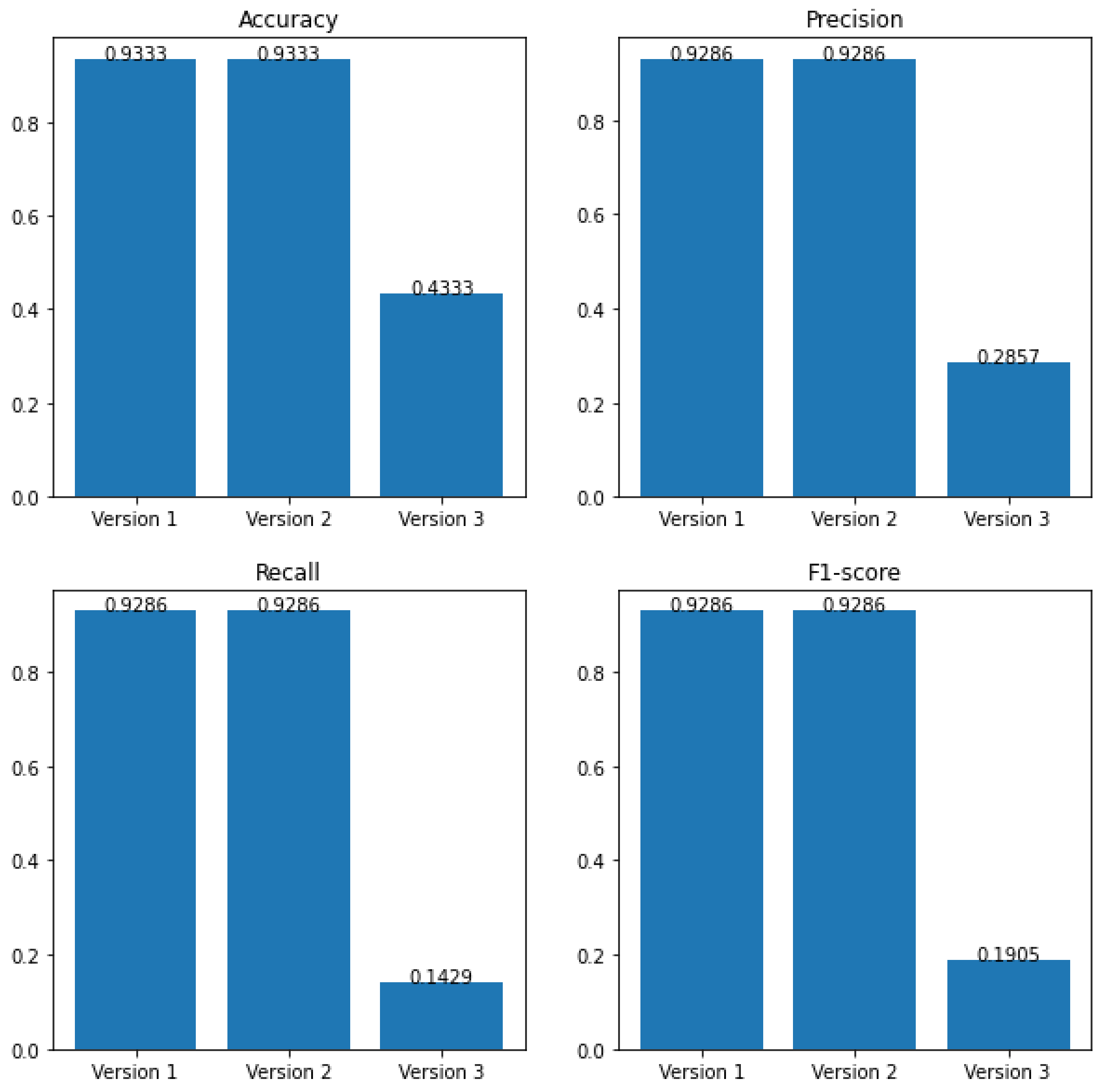


Comparing Accuracy, Precision, Recall, and F1-Score Across Versions

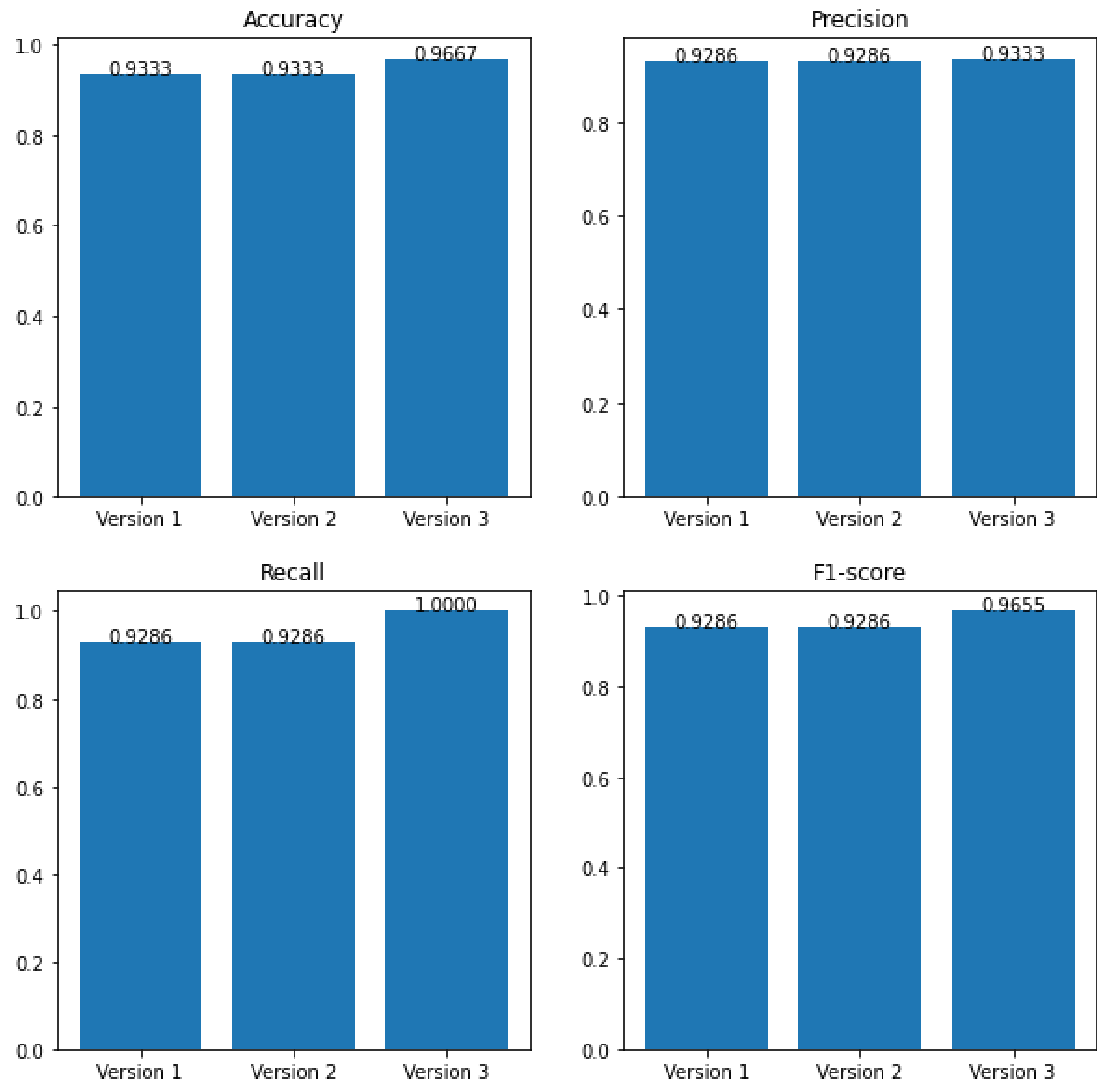
Logistic Regression Classifier



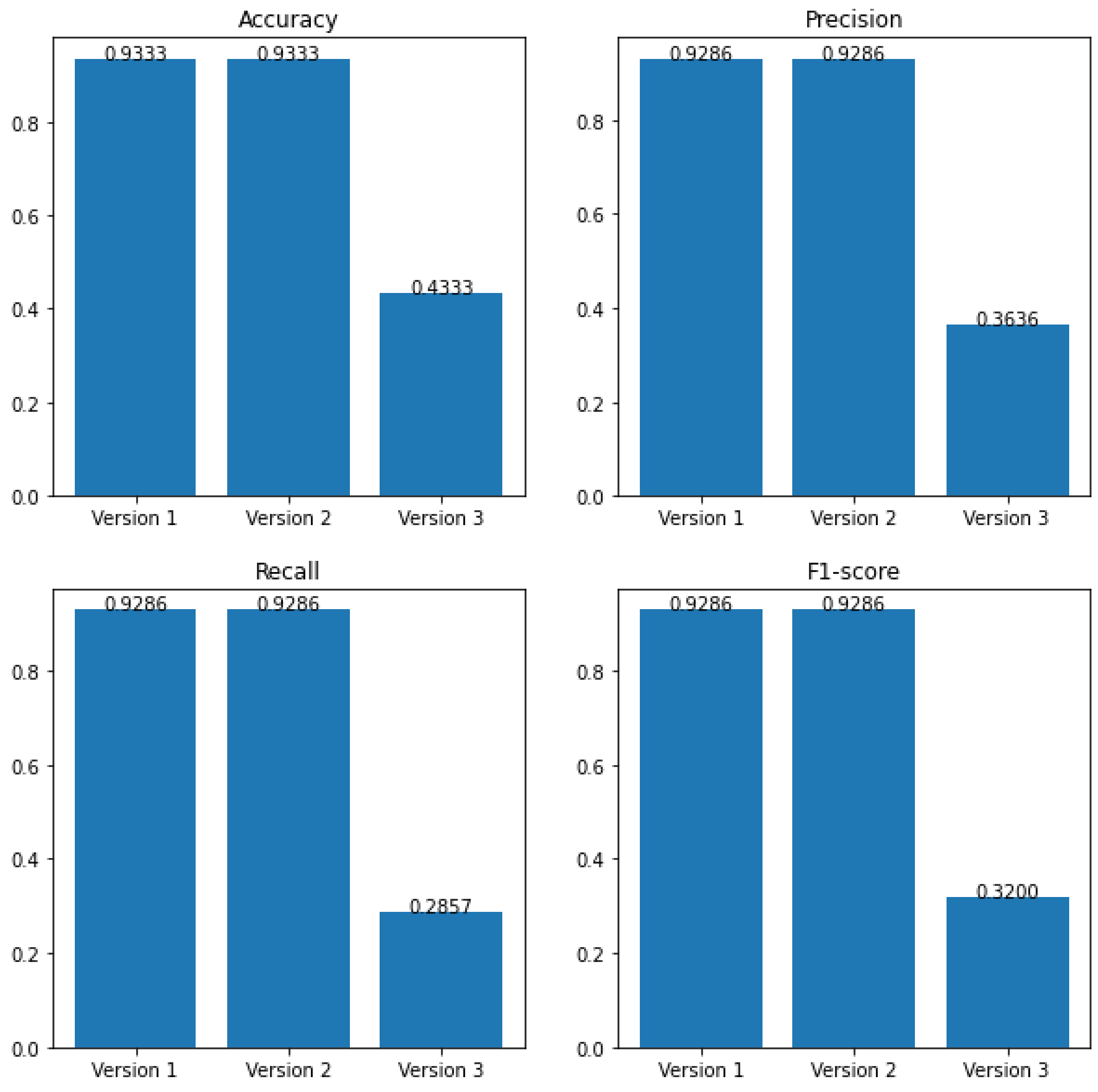
Support Vector Machine (SVM) Classifier



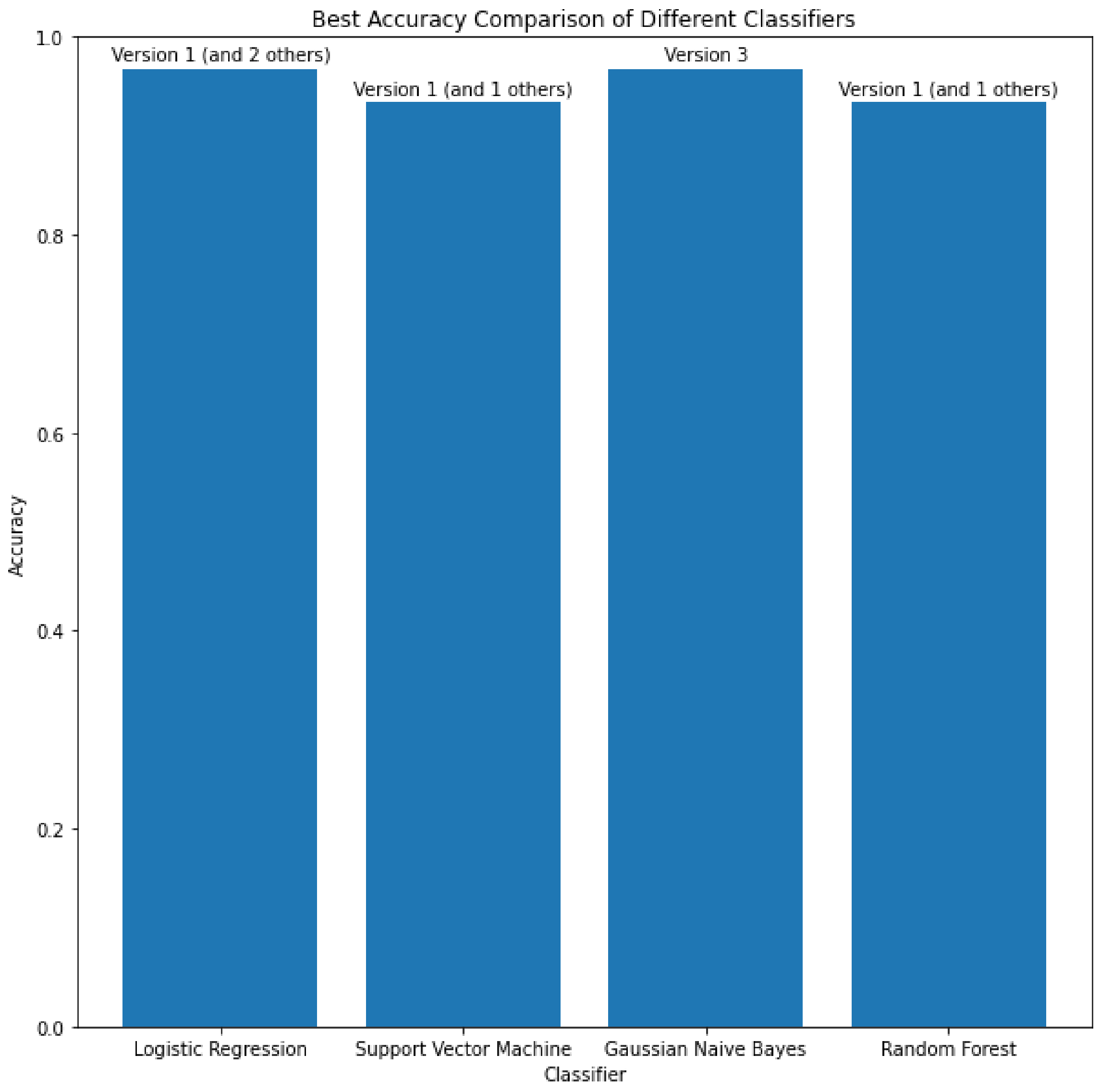
Naive Bayes Classifier

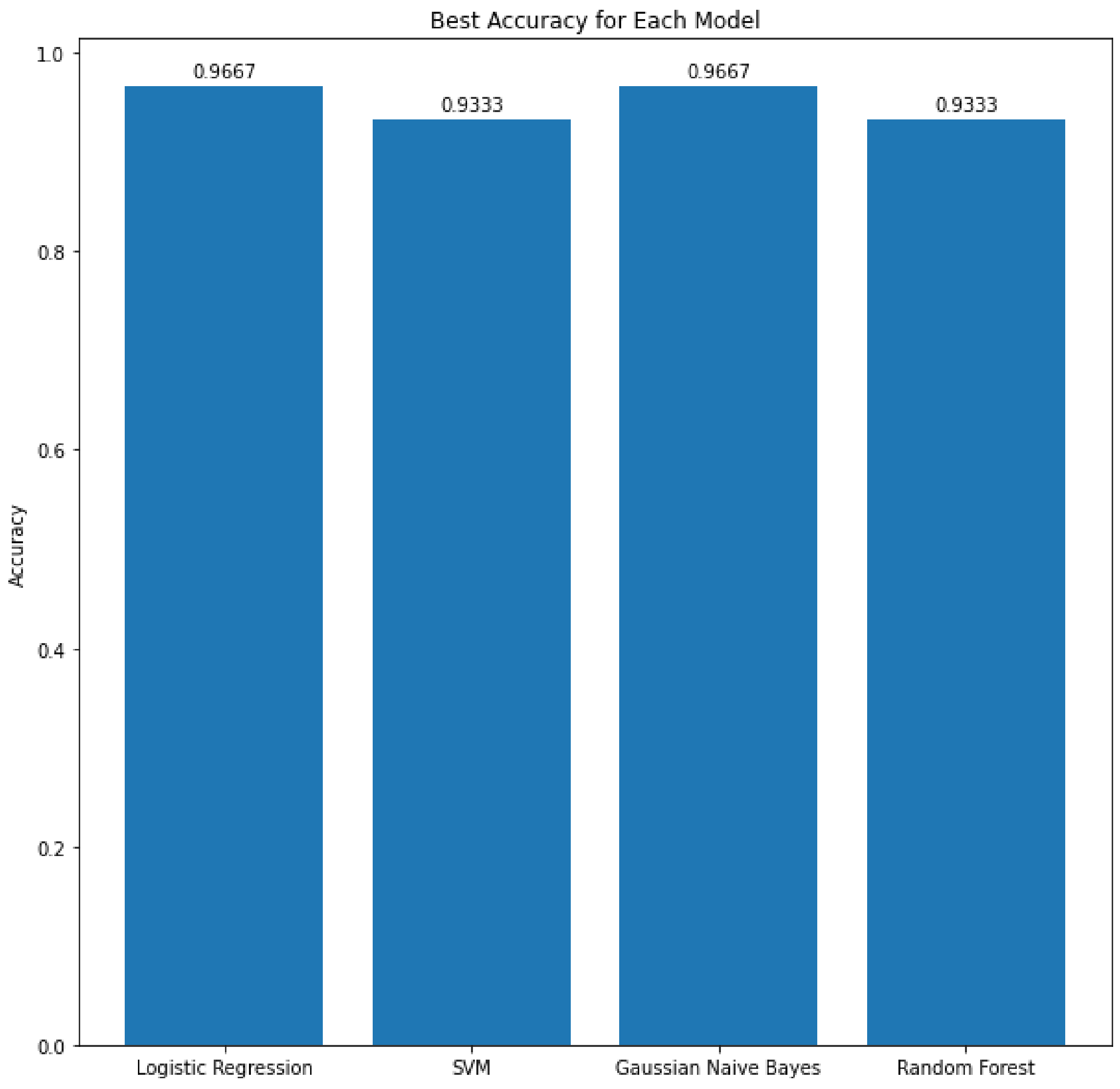


Random Forest Classifier



Best Accuracy Comparison of Different Classifiers Across Version




Best Accuracy for Each Model
Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
0.9667
0.9333
0.9667
0.9333
Logistic Regression
SVM
Gaussian Naive Bayes
Random Forest

4) Discussion

Since I created a variety of versions I was able to maximize my possible results for my model in a way that provided high statistics across the board. Each version failed for a certain model or statistic which enabled another to perform better in its stead. In the SVM and Random Forest Classifier, Version 3 underperforms across all the metrics, but every other version performs above 0.90 for all four models. Once I achieved this result, I knew that they were performing well enough.

I experienced challenges with understanding possible parameters to improve each model and different ways to process data to get improved results. I went with creating three different versions to utilize the SMOTE() function from the example code, but in addition, I wanted to experiment to find a method that could do worse and better. I spent extra time researching the models, looking at different data processing techniques, and understanding the code from the lecture to better understand it. I thought this assignment was interesting, but I think the amount of data provided should have been much larger for both sections. This would enable better training and overall results. It was exciting to dive into Machine Learning and I am ready for what is next!

5) Appendix

- GitHub
- *Google Collaboratory Code*