Classification and Prediction Model Report

Names: Luigi T. Francisco & Raven Charles Roy P. Jacinto

Course & Section: CPE 019 - CPE32S3

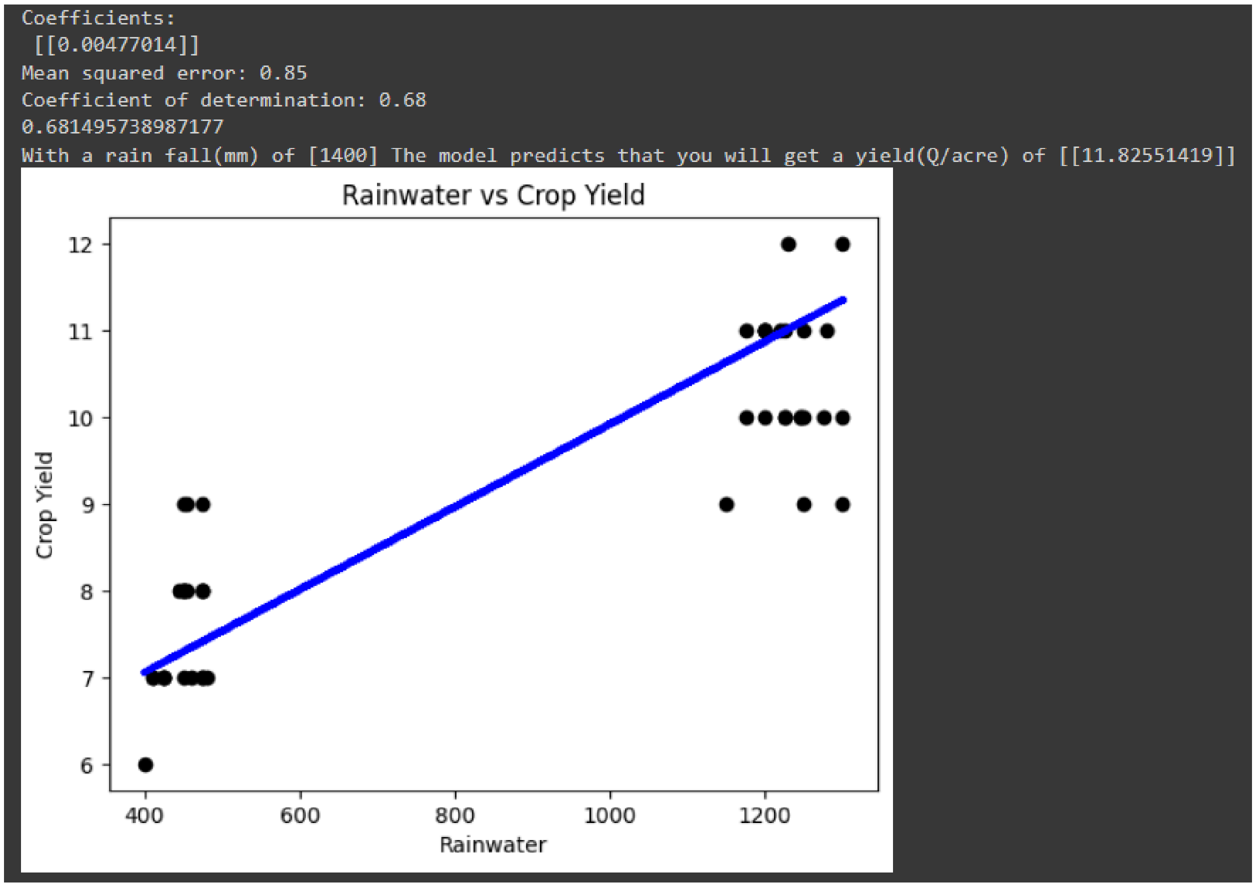
Linear Regression Models

1.) Singular Linear Regression Model

In creating the Linear Regression Model the programmers used the libraries **scikit** for machine learning and **matplotlib** for data visualization with **numpy and pandas** to load the dataset. The chosen data for our model to fit a regression line was the crop.csv containing 99 entries with columns such as rainfall, fertilizer, temperature, etc.

In training our model we split the data set between train and test with 45 entries in the train and the rest in the test. Albeit that might be weird, **the train set should normally be larger than the test set.** After that, the team chose the **rainfall to be the X variable** to be the predictor of the **Y variable(Yield (Q/acre)).** Then, it was fitted with the function fit to be used on the test variable with a coefficient of determination of 68%. As per general convention, if the coefficient of determination is greater than 60%, the X variable affects the Y variable significantly.

Figure 1.1 shows the visualization of the regression line across the scatter plot and an example prediction with the regression function having an input of 1400 in rainfall yielding 11.825 crops per acre.



```
Coefficients:
 [[0.00477014]]
Mean squared error: 0.85
Coefficient of determination: 0.68
0.681495738987177
With a rain fall(mm) of [1400] The model predicts that you will get a yield(Q/acre) of [[11.82551419]]
```

Figure 1.1 Singular Linear Regression Output

2.) Multiple Linear Regression Model

Similar to the aforementioned singular linear model, the same libraries were used except for the **matplotlib3D**. The team here put multiple features into the **X** variable(A data frame) with two columns namely the rainfall and fertilizer count to be fitted against the Y variable (Yield(Q/acre)). It goes the same process with the **scikit** with the model being an object of the linear model function though with a different visualization process and the number of coefficients to be calculated with the X variables and the determined intercept. A difference is the data split with train data having 70% of the entries and 30% for the test data.

The model the team created has an accuracy of 77% for the test data and 79% for the whole data with a coefficient of determination to be 77%. It is considered for the features X(rainfall and fertilizer) to be significantly affecting the yield of crops.

Figure 1.2 shows the visualization of Multiple Linear Regression with matplotlib toolkit for 3D and Figure 1.3 is an example data with arbitrary rainfall and fertilizer predicted yield also the coefficients and intercept for the regression function along with the accuracy of the regression model.

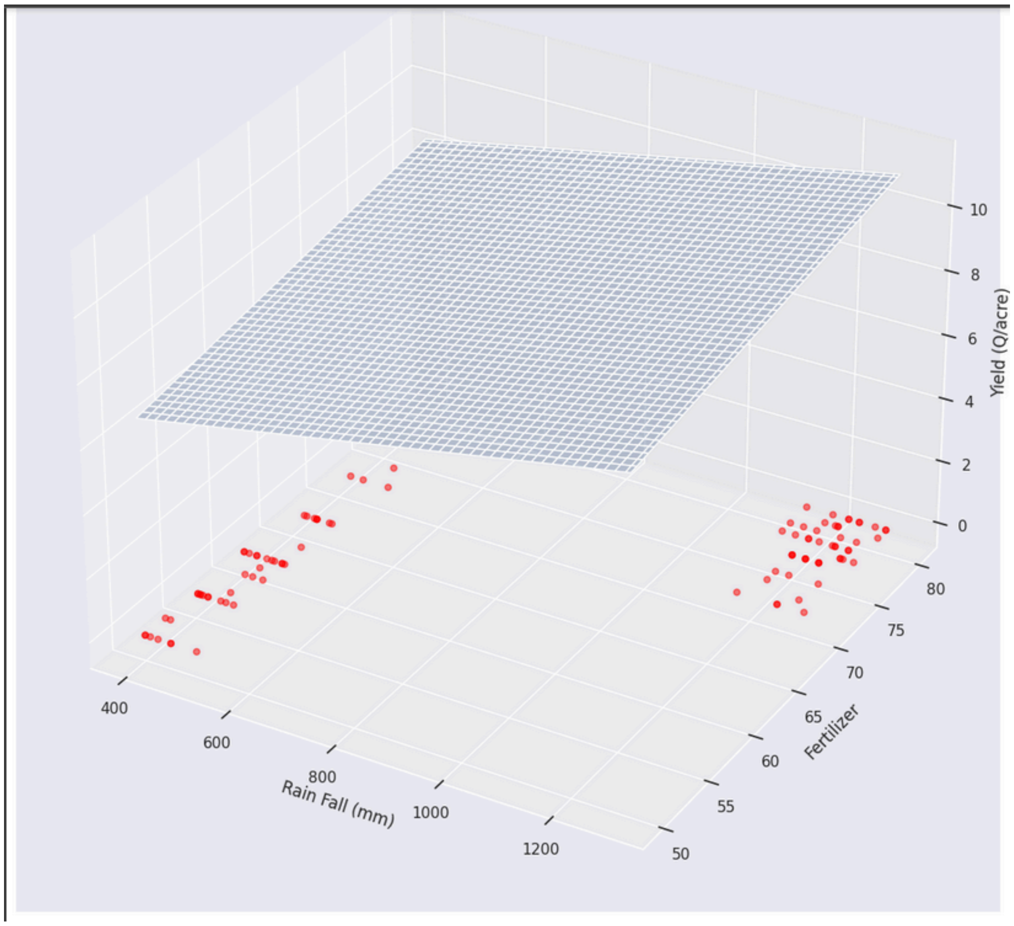
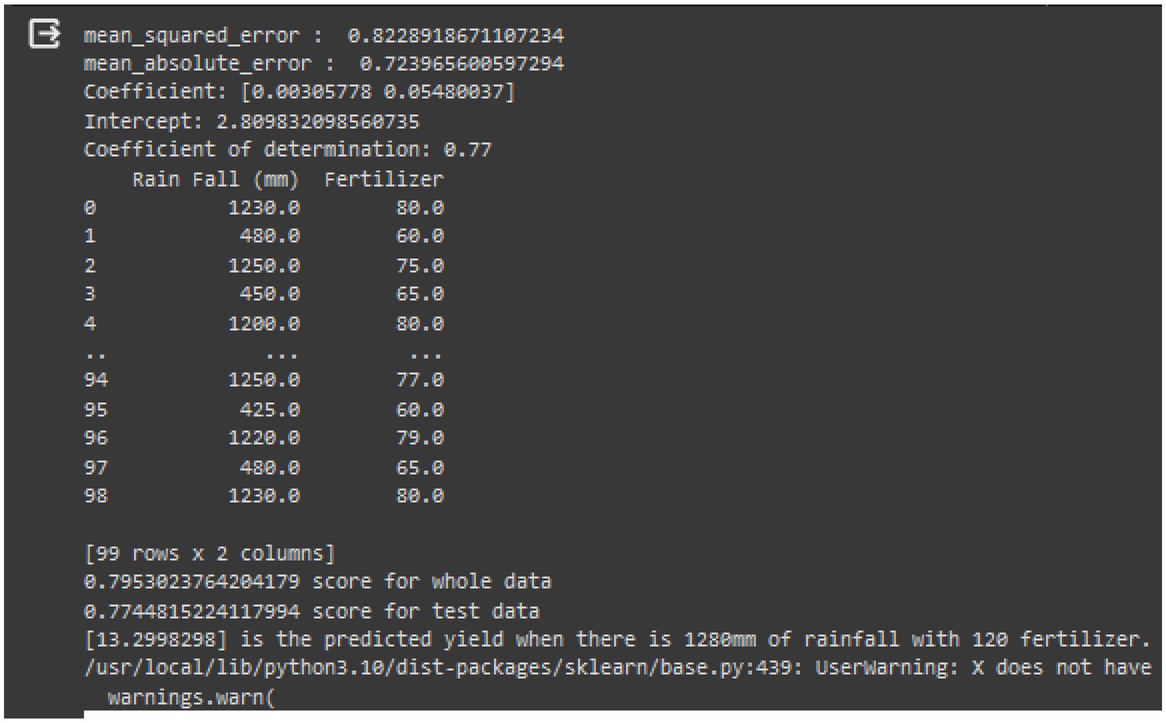


Figure 1.2 Multiple Linear Regression 3D



```
mean_squared_error :  0.8228918671107234
mean_absolute_error :  0.723965600597294
Coefficient: [0.00305778 0.05480037]
Intercept: 2.809832098560735
Coefficient of determination: 0.77
    Rain Fall (mm)  Fertilizer
0           1230.0        80.0
1            480.0        60.0
2           1250.0        75.0
3            450.0        65.0
4           1200.0        80.0
..             ...         ...
94          1250.0        77.0
95           425.0        60.0
96          1220.0        79.0
97           480.0        65.0
98          1230.0        80.0

[99 rows x 2 columns]
0.7953023764204179 score for whole data
0.7744815224117994 score for test data
[13.2998298] is the predicted yield when there is 1280mm of rainfall with 120 fertilizer.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have
  warnings.warn(
```

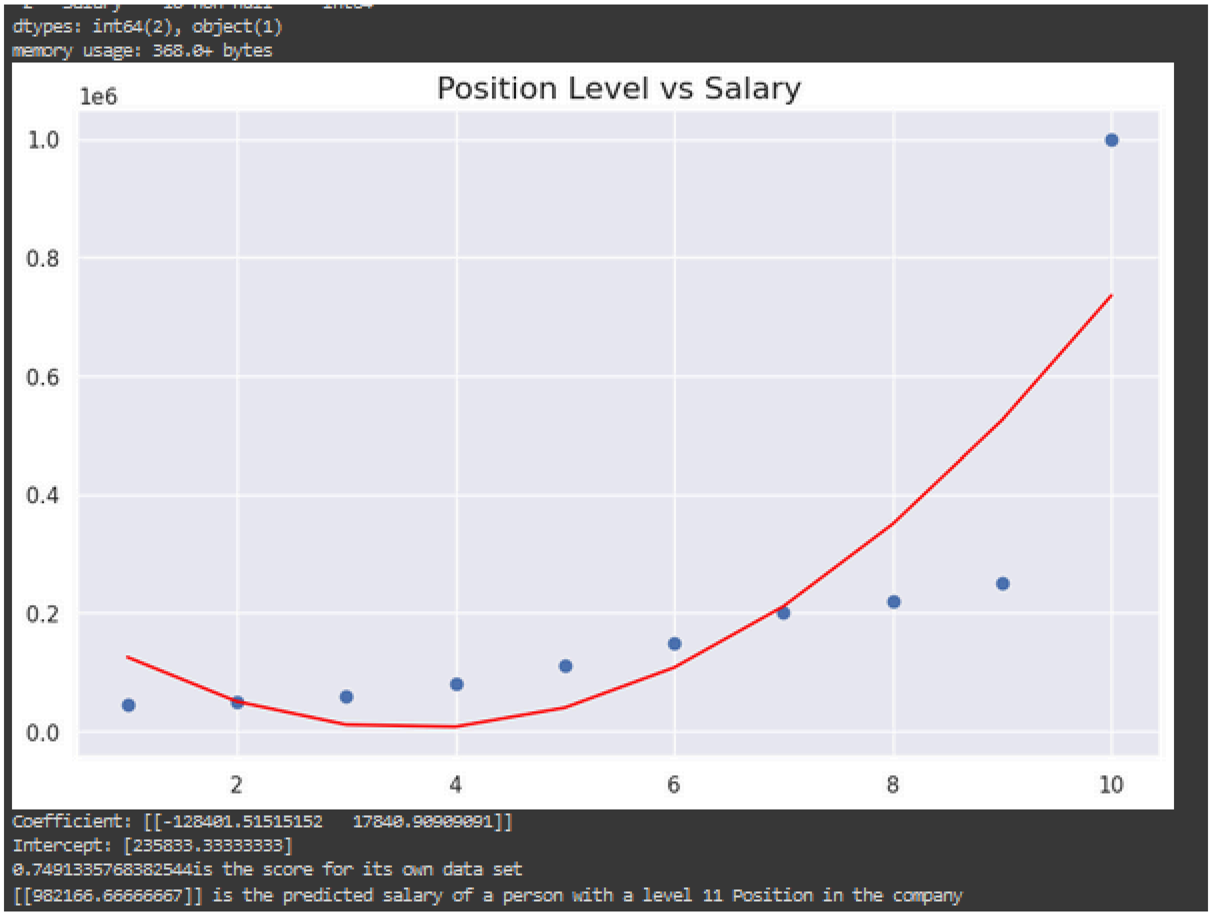
Figure 1.3 Prints for Multiple Linear Regression

3.) Polynomial Regression Mode

In comparison to the prior regression models, Polynomial Regression requires the user to view the scatter plot to assume what kind of polynomial feature it fits. The dataset here is different from the prior one as there was no observable polynomial relationship between the variables in the crop data set. As such a dataset with position, level , and salary was taken with 10 entries. As there were too few entries. The team did not separate it to train and test the data set.

In implementing polynomial regression it is required for the X variable to be poly-transformed to expand the data set with its corresponding polynomial value. An example is the X variable being squared as such the variable X data frame is expanded with the transformed value. The team then set up a model object for the linear model to be fitted similar to the singular regression method. The accuracy of the model against its own data set is 76% which seems to indicate a good fit with a polynomial degree of 2.

Figure 1.4 shows the visualization for the polynomial regression and the printed accuracy of the model against its own data set with an example of a prediction for the salary of a level 11 position in the company.


```
dtypes: int64(2), object(1)
memory usage: 368.0+ bytes
```

```
Coefficient: [[-128401.51515152   17840.90909091]]
Intercept: [235833.33333333]
0.7491335768382544is the score for its own data set
[[982166.66666667]] is the predicted salary of a person with a level 11 Position in the company
```

Figure 1.4 Polynomial Regression Output

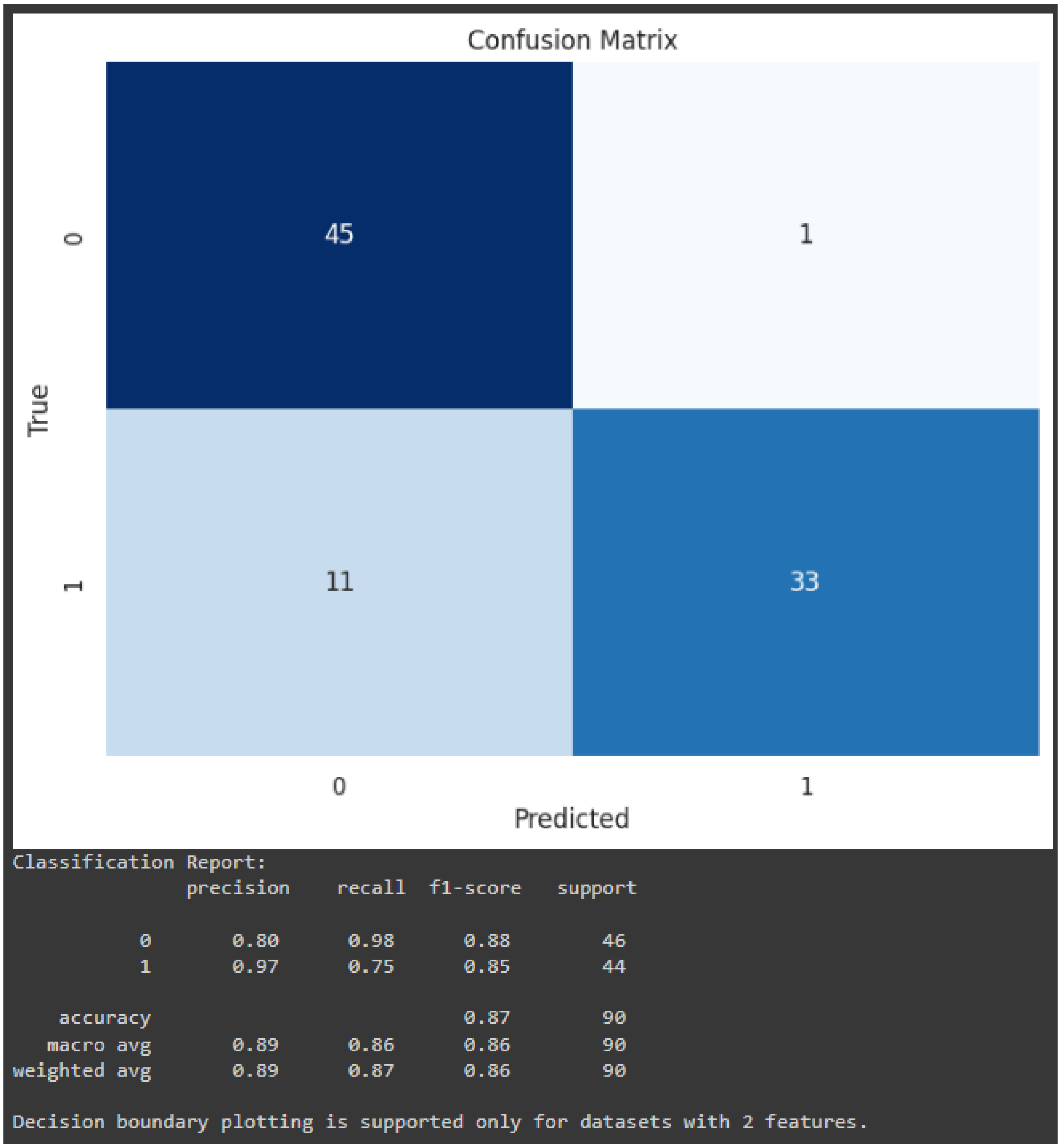
Logistic Regression Model

In creating the Logistic Regression Model the programmers used the libraries **pandas** to load the dataset. **Scikit/sklearn** for machine learning and **matplotlib**, **seaborn**, and **numpy** for data visualization. The chosen data for our model to fit a regression line was the HeartDiseaseData.csv containing 303 entries with 15 columns such as Age, Sex, ChestPain, RestBP, Chol, Fbs, RestECG, MaxHR, ExAng, Oldpeak, Slop, Ca, Thal, and AHD. Since the data needs a dichotomous answer/prediction whether the person is having heart disease or not based on the predictors/independent variables data that we have.

In training our model we organize the data since some of the data is bigger than other, it would be overpowering for the other so we manage that using a scaler that uses mean and STD to scale the data. We then split the data set between train and test with some entries in the train and the rest in the test. After that, the team chose **all the columns except the AHD column to be the X variable (input variables)** to be the predictor of the **Y variable/Target Value (AHD).**

As per general convention, if the accuracy of the training data is close to the accuracy of test data, the model is working correctly and appropriately that can be measured by the accuracy score. Therefore, the model can predict whether the person has acquired heart disease (AHD) based on the data we feed on the model.

Figure 1.5 shows the visualization of the confusion matrix classification report for the logistic regression. Wherein it visualizes the true positive, true negative, false positive, and false negative values. This helps to visualize how the model separates the different classes in feature space. 45 is true negative, 33 is true positive, 1 false negative, 11 is false positive.



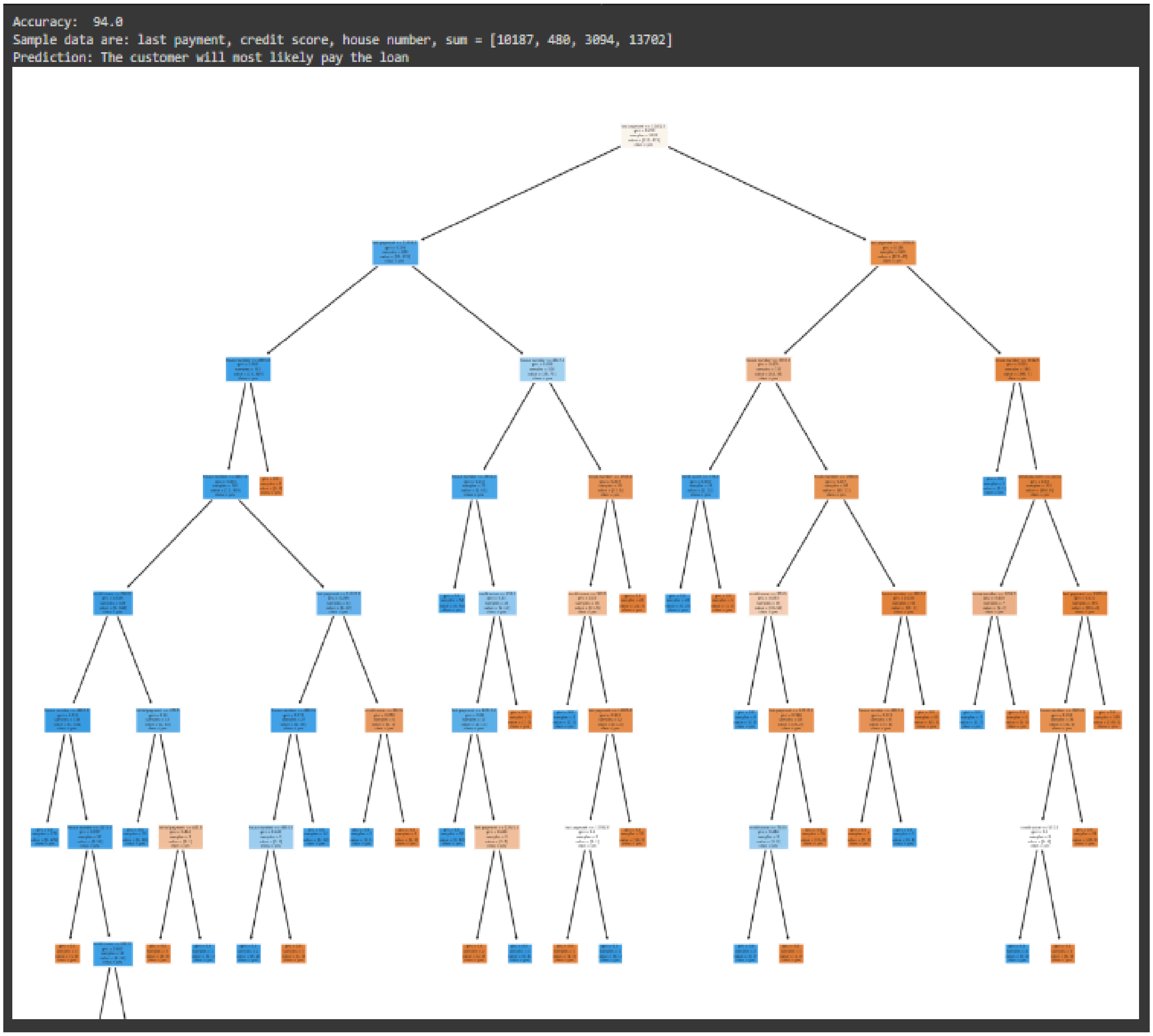
```
Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.98      0.88        46
           1       0.97      0.75      0.85        44

    accuracy                           0.87        90
   macro avg       0.89      0.86      0.86        90
weighted avg       0.89      0.87      0.86        90

Decision boundary plotting is supported only for datasets with 2 features.
```

Figure 1.5 Logistic Regression Output

Decision Tree Model

In performing the Decision Tree Model, our team discovered that its function is to solve classification problems and to categorize objects based on their characteristics. It can be used to predict both categorical and discrete target values. We have loaded that data named Bank_data.csv wherein 1000 rows/entries and 6 columns can be found. In this case, we want to predict whether the customer of the bank will pay their loan or not based on their initial payment, last payment, credit score, house number, sum. Our data is more on discrete values and still we want to predict the results that is why we use the word yes and no for it. Yes meaning the bank customer will pay the loan and no means will not pay the loan..

In training our data set, what we did is we slightly organized our data and then proceeded to separate our target value (y) and independent variables (x) to each other so that we can use it in training models. Next is, we use train_test_split to train the model on one set and evaluate its performance on another set. Then we created a decision tree classifier using the entropy criterion, limiting the maximum depth of the tree to 3 and setting a minimum number of samples required at a leaf node to 5. Then, it trains the classifier on the provided training data. Next to that is the prediction based on entropy that we get on our training or test data. This prediction will give us the results of yes or no to solve our problem. Lastly, we visualize the result of the tree using plot tree from sklearn and that can be seen in figure 1.6.



```
Accuracy:  94.0
Sample data are: last payment, credit score, house number, sum = [10187, 480, 3094, 13702]
Prediction: The customer will most likely pay the loan
```

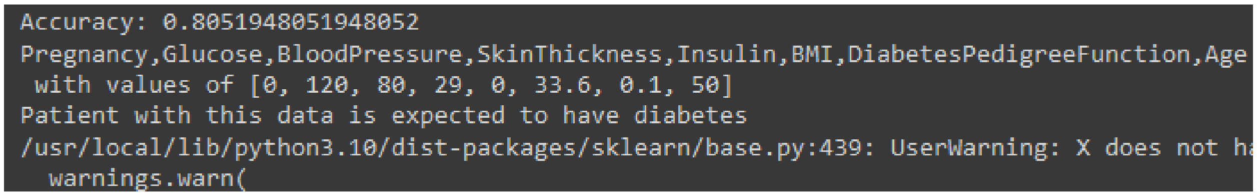
Figure 1.6 Plot Tree of the Decision Tree Model

Random Forest Model

In performing Random Forest, the team found it applicable in classifying if a person has diabetes or not as such a data set with regards to diabetes had been used for the model. The team created an X and Y variable where variable X is a data frame composed of all the columns of the original data frame except the column regarding Y which is the outcome if a person has diabetes or not.

After that, The data set is divided further into train and test sets. The team created an object to hold the RandomForestClassifier where it fits the train data set. The default of 100 trees in the forest in our classifier was used to predict the test data where the accuracy varies by iteration. Although there is a variance in accuracy it is generally within the range of 60%-90% as such it can be said that the model is accurate and significant.

Figure 4.1 shows the accuracy of the model on one of its iterations and an example data for prediction where it classifies if a person has diabetes. Figure 4.2-4.6 shows five trees of the 100 trees generated by the model to classify if a person has diabetes. Observing those trees have different order of attributes used for classification due to the differing data set used for each tree with bootstrapping.



```
Accuracy: 0.8051948051948052
Pregnancy,Glucose,BloodPressure,SkinThickness,Insulin,BMI,DiabetesPedigreeFunction,Age
 with values of [0, 120, 80, 29, 0, 33.6, 0.1, 50]
Patient with this data is expected to have diabetes
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not ha
  warnings.warn(
```

Figure 4.1 Accuracy and Example Data for Classification Output

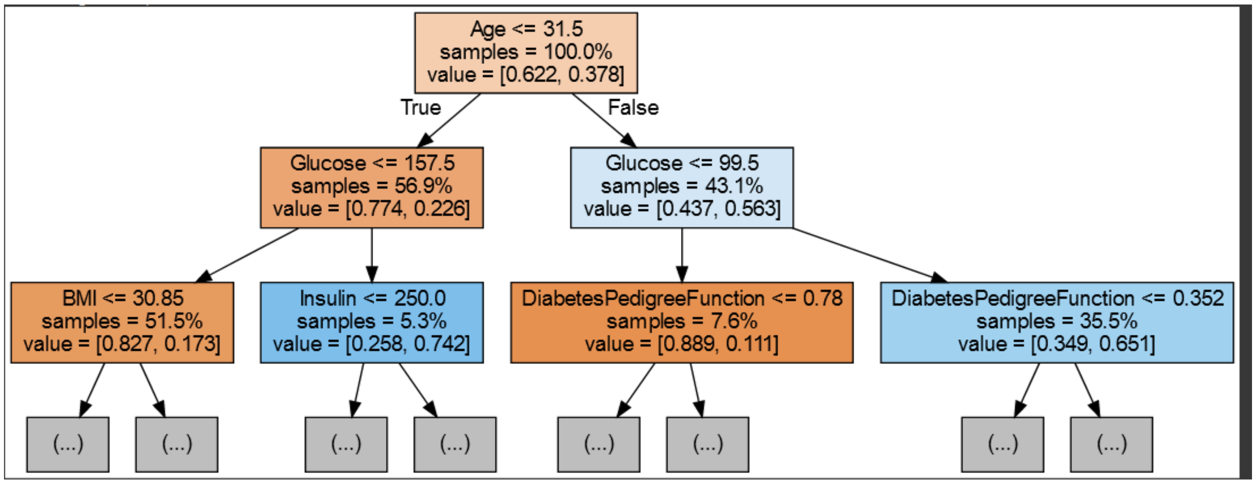


Figure 4.2 1st Tree Generated

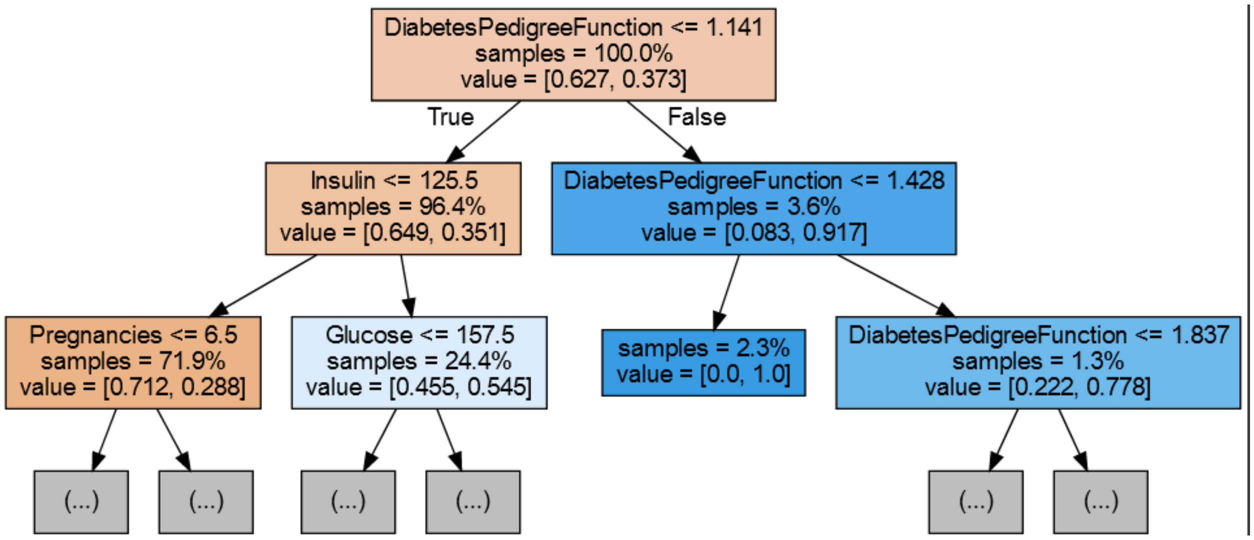


Figure 4.3 2nd Tree Generated

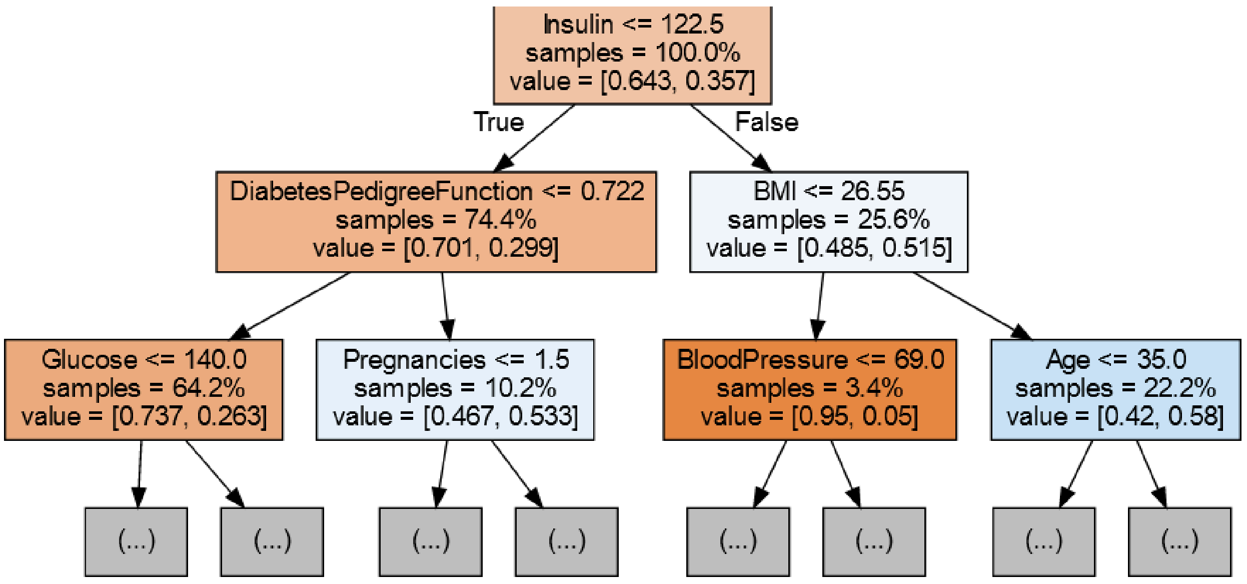


Figure 4.4 3rd Tree Generated

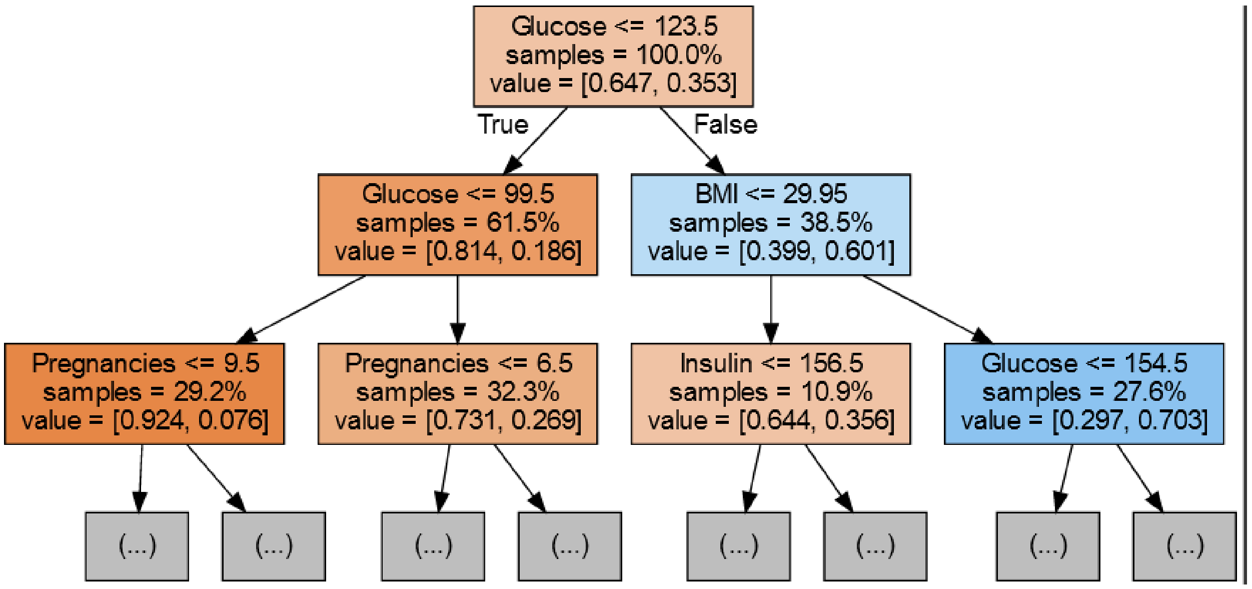


Figure 4.5 4th Tree Generated

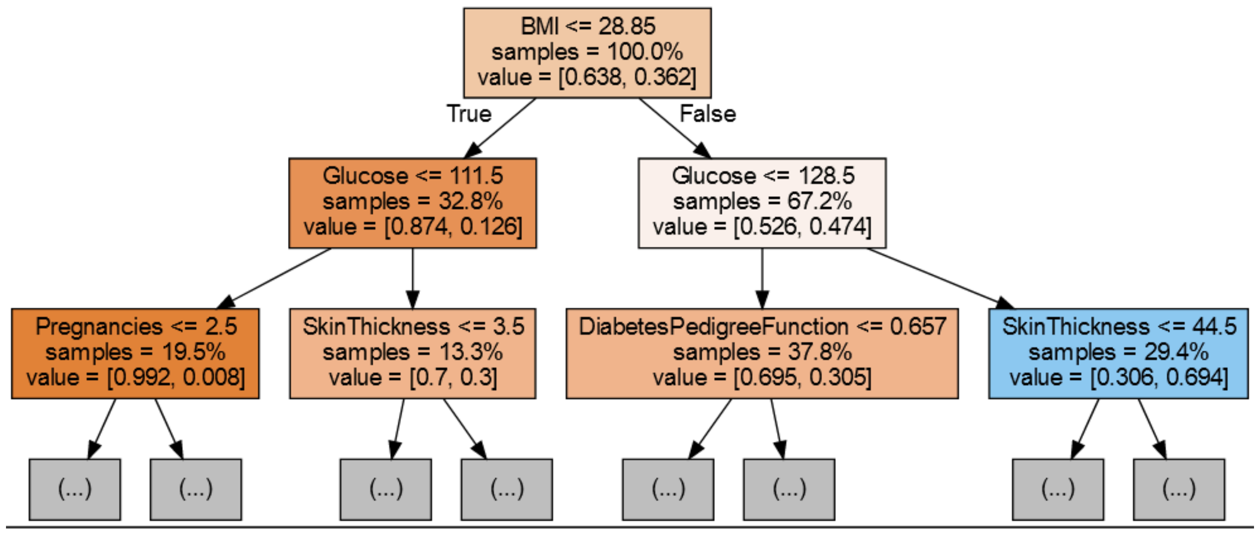


Figure 4.6 5th Tree Generated