Enterprise Cloud Computing and Big Data (BUDT737)

Prediction of Breast Cancer Risk

Team Members: Poojaa Ravi Kumar, Xinming Tan, Helena Getachew

ORIGINAL WORK STATEMENT

We the undersigned certify that the actual composition of this proposal was done by us and is original work.

	Typed Name	Signature
	Poojaa Ravi Kumar	*Poojaa Ravi Kumar*
	Xinming Tan	*Xinming Tan*
	Helena Getachew	*Helena Getachew*

Executive Summary

Our project on predicting breast cancer through machine learning effectively utilized various classification techniques to achieve precise and early identification of breast tumors. With the combination of advanced data analytics, optimization, and evaluation methods, the purpose of this project was to predict whether a breast mass was malignant or benign based on 32 characteristics. Key findings in this report include the average dimensions of malignant and benign tumors. In addition, the accuracies of three models- logistic regression, decision tree, and random forest- were compared, and the logistic regression was found to have the highest accuracy. Overall, this project achieved significant strides in predicting the likelihood of malignancy based on a set of well-defined features extracted from breast cancer data.

Data Description:

Data source: https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic
Sample size: 569 recordings and 32 attributes - 18208

About the data:

The data used in this project is sourced from diagnostic data collected by Dr. William H. Wolberg from the University of Wisconsin-Madison in 1990. Images of Fine Needle Aspirations (FNA) of 569 breast masses were analyzed across several characteristics. Each FNA sample contains cells, tissues, and fields. Measurements of the nucleus within each cell can be collected to make a diagnosis. The aim of the data set was to utilize machine learning techniques to diagnose these masses as either malignant or benign based on nine features:

radius: distance between center to nucleus perimeter (mm)
texture: standard deviation of gray-scale values (mm)
perimeter: distance around the nucleus (mm)
area: distance across the surface of the nucleus (mm^2)
smoothness: local variation in radius lengths
compactness: perimeter^2/area - 1.0
concavity: size of the concave regions in the nucleus
concave points: number of concave regions in the nucleus
symmetry: how identical the nucleus is on either side
fractal dimension: coastline approximation – 1

The mean, standard error, and "worst" (largest) measure of each of the nine features were recorded in this data set. Each record has a unique id and diagnosis ('M'=malignant or 'B'= benign). In total, there are 32 columns.

Sample observations:

id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
842302	M	17.99	10.38	122.8	1001	0.1184
842517	M	20.57	17.77	132.9	1326	0.08474
84300903	M	19.69	21.25	130	1203	0.1096

Why is the data important?

Breast cancer continues to be one of the most diagnosed cancers in the United States. According to the Center for Disease Control, around 240,000 women and 2,100 men are diagnosed with breast cancer each year. Additionally, breast cancer claims the lives of 42,000 women and 500 men each year in the U.S. This data is of interest because it investigates the predictive power of diagnostic tools used to distinguish between cancerous and non-cancerous tumors.
It is essential to note that not all tumors are cancerous; they can be classified as:

- Benign (non-cancerous)
- Pre-malignant (pre-cancerous)
- Malignant (cancerous)

Diagnostic procedures such as MRI, mammogram, ultrasound, and biopsy are commonly employed to identify and assess breast cancer.

Research Questions:

1. What are the average dimensions for a malignant mass in this dataset?
2. What are the average dimensions for a benign mass?
3. Between logistic regression, decision tree, and random forest models, which model is most accurate for correctly predicting the diagnosis of each mass?

Methodology:

The analysis is divided into four sections:

Identifying the problem and Data Source

First, the problem was identified, and we imported external data sets to understand and get familiarized with the information to think of different ways to handle the data.

Expected Outcome:

The breast cancer diagnosis results in this dataset comes from breast fine needle aspiration (FNA) test (is a quick and simple procedure to perform, which removes some fluid or cells from a breast lesion or cyst (a lump, sore or swelling) with a fine needle like a blood sample needle).

Since this build a model that can classify a breast cancer tumor using two training classification:

1= Malignant (Cancerous) - Present
0= Benign (Not Cancerous) -Absent

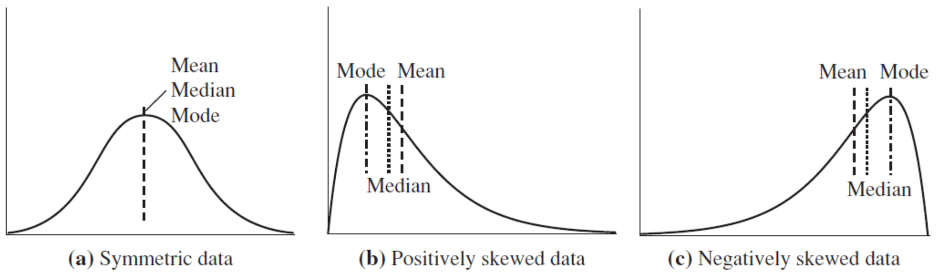
Exploratory Data Analysis

Here, we understand the patterns present within the data and the data itself by using data exploration and visualization techniques by importing python libraries (Pandas, matplotlib and seaborn) into spark. This will help us assess how the features relate to the target variable. Knowledge and familiarity with the data is crucial to be able to provide useful information. This information will be used for data pre-processing.

Purpose of EDA:

- to use summary statistics and visualizations to better understand data,
- find clues about the tendencies of the data, its quality and to formulate assumptions and the hypothesis of our analysis.
- For data preprocessing to be successful, it is essential to have an overall picture of our data.

We have used summary statistics to determine any null values and removed them from the dataset. The column id was also removed as the unique nature of it has no predictive power for the dataset and hence is not useful. We also employed skewness to understand the tendency of all the features.



.I Mean, median, and mode of symmetric versus positively and negatively skewed data.

The skew result shows a positive (right) or negative (left) skew. Values closer to zero show less skew. In the context of this project, features having significant skewness may exhibit distinct groupings between malignant and benign cancer types. The mentioned features could be valuable in differentiating between the two cancer types due to their distribution characteristics.

The features radius_mean, perimeter_mean, area_mean, concavity_mean and concave_points_mean is useful in predicting cancer type due to the distinct grouping between malignant and benign cancer types in these features.

Feature analysis through visualization

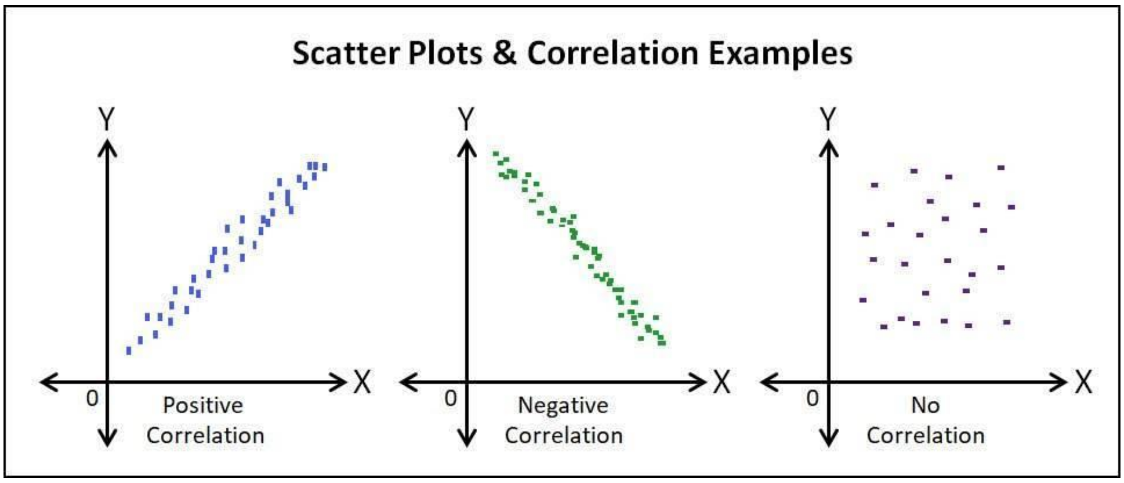
Visual displays of statistical descriptions. In this project, we have employed two types.

1. **Univariate**

   The visualization used is a histogram to display the numeric data. Through this, a few features with exponential distribution were identified. This is useful for model selection as many machine learning techniques are sensitive to distribution where it assumes normal distribution. For our case, it may be useful to consider algorithms like Decision Trees, Random Forests, and Neural Networks or perform standardization of data. See Figures A-C for a display of the histograms.

2. **Bivariate**

   The visual involving two attributes used here is a correlation matrix to determine the relationship between two numeric variables. Two attributes, X, and Y are correlated if one attribute implies the other. Correlations can be positive, negative, or null (uncorrelated)



Pre-Processing the Data

The target variable which is categorical is label encoded to have a binary value which makes it easier for the models to interpret. The dataset is then split to train and test data. Both these datasets are now standardized.

Furthermore, it is also possible to perform feature selection to reduce high dimension data which will improve the performance of the models. The features with the highest predictive power can be filtered and can be used to train predictive models.

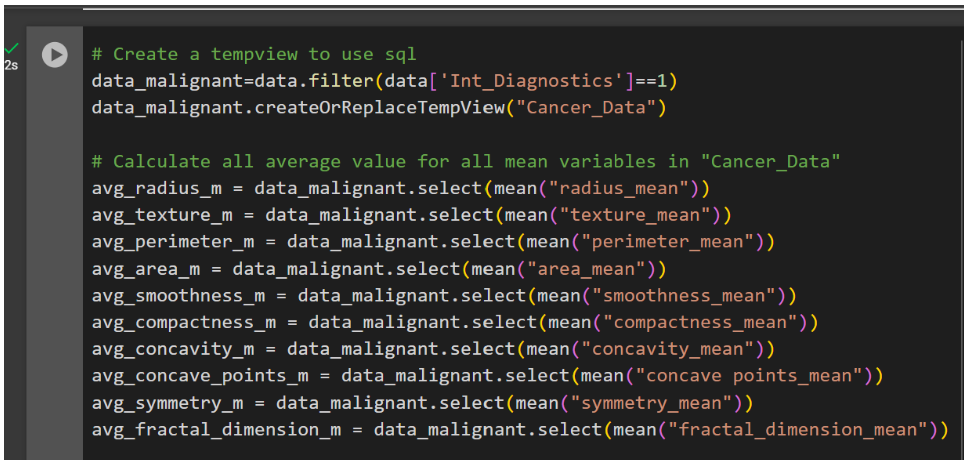
Build a model to predict whether breast cell tissue is Malignant or Benign.

Here, different predictive models are constructed, and their accuracies are compared. The SVM machine learning algorithm had the best accuracy to predict diagnosis of a breast tumor. The model is also evaluated using confusion matrix, which is essential to assess and interpret the fitted model. Since it is important to bring the False Negative Rate to 0, it is possible to tune the model such that FNR is decreased. But that would also mean the False Positive Rate would be increased.

Results and Findings:

1. The average dimensions of a malignant tumor

Leveraging SQL techniques in PySpark, the data frame was filtered so that it only included the rows for masses diagnosed as malignant (‘M’). Then, the averages of each of the 10 features were determined using the following queries:



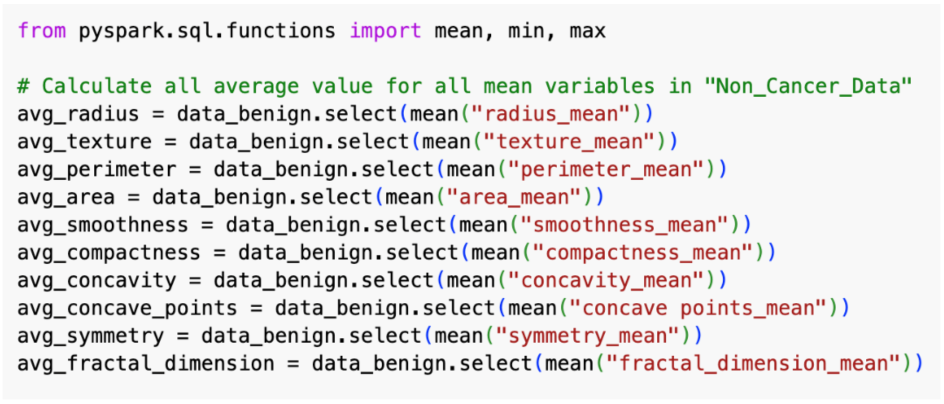
```
# Create a tempview to use sql
data_malignant=data.filter(data['Int_Diagnostics']==1)
data_malignant.createOrReplaceTempView("Cancer_Data")

# Calculate all average value for all mean variables in "Cancer_Data"
avg_radius_m = data_malignant.select(mean("radius_mean"))
avg_texture_m = data_malignant.select(mean("texture_mean"))
avg_perimeter_m = data_malignant.select(mean("perimeter_mean"))
avg_area_m = data_malignant.select(mean("area_mean"))
avg_smoothness_m = data_malignant.select(mean("smoothness_mean"))
avg_compactness_m = data_malignant.select(mean("compactness_mean"))
avg_concavity_m = data_malignant.select(mean("concavity_mean"))
avg_concave_points_m = data_malignant.select(mean("concave points_mean"))
avg_symmetry_m = data_malignant.select(mean("symmetry_mean"))
avg_fractal_dimension_m = data_malignant.select(mean("fractal_dimension_mean"))
```

Table 1 includes the output of the above code. The average dimensions are as follows:

- Radius: 17.46 mm
- Texture: 21.60 mm
- Perimeter: 115.36 mm
- Area: 978.36 mm^2
- Smoothness: 0.10
- Compactness: 0.14
- Concavity: 0.160
- Concave Points: 0.09
- Symmetry: 0.19
- Fractal Dimension: 0.06

2. The average dimensions of a benign tumor

Again, by using PySpark with SQL, the data frame was filtered so that it only included the rows for masses diagnosed as benign (‘B’). Then, the averages of each of the 10 features were determined using the following queries:



```
from pyspark.sql.functions import mean, min, max

# Calculate all average value for all mean variables in "Non_Cancer_Data"
avg_radius = data_benign.select(mean("radius_mean"))
avg_texture = data_benign.select(mean("texture_mean"))
avg_perimeter = data_benign.select(mean("perimeter_mean"))
avg_area = data_benign.select(mean("area_mean"))
avg_smoothness = data_benign.select(mean("smoothness_mean"))
avg_compactness = data_benign.select(mean("compactness_mean"))
avg_concavity = data_benign.select(mean("concavity_mean"))
avg_concave_points = data_benign.select(mean("concave points_mean"))
avg_symmetry = data_benign.select(mean("symmetry_mean"))
avg_fractal_dimension = data_benign.select(mean("fractal_dimension_mean"))
```

Table 1 includes the output of the above code. The average dimensions are as follows:

- Radius: 12.15 mm
- Texture: 17.91
- Perimeter: 78.08 mm
- Area: 462.79 mm^2

- Smoothness: 0.09
- Compactness: 0.08
- Concavity: 0.046
- Concave Points: 0.03
- Symmetry: 0.17
- Fractal Dimension: 0.06

3. Which model is the best? Logistic Regression, Decision Tree, or Random Forest?

The machine learning package within PySpark was utilized to train and test logistic regression, decision tree, and random forest models. By splitting the data into 70% train and 30% test data, each model yielded high accuracies close to 92-94%. To reduce bias within the training sampling, cross-validation was performed. Thus, the training data was split into 5 folds and each model was trained using cross validation. Then, to address large disparities in feature quantities, min-max scaling was conducted on variables containing values of particularly large sizes. Also, standard scaling was performed on all the features to scale the data around a mean of 0 and standard deviation of 1. Using these feature processing techniques coupled with cross validation was intended to address any overfitting. See Figures D-L for a display of the feature engineering, model development, and cross validation steps as well as the actual diagnoses, predicted classes, and probabilities for the first 20 rows within the dataset.

Despite these efforts, the validation accuracy of the 3 models remained around 92-94%. However, feature engineering and cross-validation are still valuable steps to improve the generalization performance of each model. Thus, the feature processing and cross-validation steps are still meaningful in terms of demonstrating the reliability of the models.

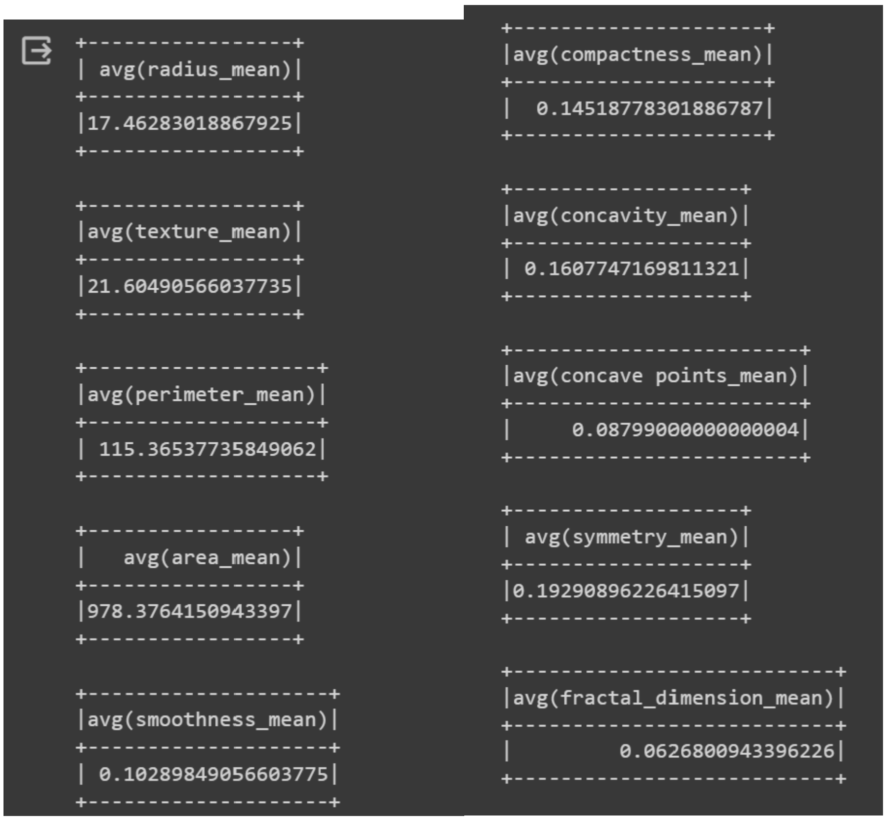
Ultimately, the logistic regression model had the highest accuracy compared to the decision tree and random forest models after feature processing and cross-validation.

Conclusion

In conclusion, our breast cancer prediction machine learning project successfully employed different classification models to determine accurate and early detection of breast tumor. The cross validation and confusion matrix analysis validated the model’s robustness. Further optimizing this model with feature selection, hyperparameter can significantly contribute to the model’s predictive performance. Despite the positive outcomes, there are few limitations such as not being able to identify all true positives in the dataset.

Appendix:

Table 1.



```
+-----------------+
| avg(radius_mean)|
+-----------------+
|17.46283018867925|
+-----------------+

+-----------------+
|avg(texture_mean)|
+-----------------+
|21.60490566037735|
+-----------------+

+-------------------+
|avg(perimeter_mean)|
+-------------------+
| 115.36537735849062|
+-------------------+

+-----------------+
|   avg(area_mean)|
+-----------------+
|978.3764150943397|
+-----------------+

+--------------------+
|avg(smoothness_mean)|
+--------------------+
| 0.10289849056603775|
+--------------------+
```

```
+---------------------+
|avg(compactness_mean)|
+---------------------+
|  0.1451877830188678 7|
+---------------------+

+-------------------+
|avg(concavity_mean)|
+-------------------+
| 0.1607747169811321|
+-------------------+

+------------------------+
|avg(concave points_mean)|
+------------------------+
|     0.08799000000000004|
+------------------------+

+-------------------+
| avg(symmetry_mean)|
+-------------------+
|0.19290896226415097|
+-------------------+

+---------------------------+
|avg(fractal_dimension_mean)|
+---------------------------+
|         0.0626800943396226|
+---------------------------+
```

Table 2

```
+------------------+
| avg(radius_mean)|
+------------------+
|12.14652380952381|
+------------------+

+-------------------+
| avg(texture_mean)|
+-------------------+
|17.914761904761892|
+-------------------+

+--------------------+
|avg(perimeter_mean)|
+--------------------+
|  78.07540616246497|
+--------------------+

+-------------------+
|    avg(area_mean)|
+-------------------+
|462.79019607843145|
+-------------------+

+---------------------+
|avg(smoothness_mean)|
+---------------------+
| 0.09247764705882354|
+---------------------+
```

```
+---------------------+
|avg(compactness_mean)|
+---------------------+
|  0.08008462184873952|
+---------------------+

+-------------------+
|avg(concavity_mean)|
+-------------------+
|0.04605762100840336|
+-------------------+

+------------------------+
|avg(concave points_mean)|
+------------------------+
|    0.025717406162464995|
+------------------------+

+------------------+
|avg(symmetry_mean)|
+------------------+
|0.1741859943977591|
+------------------+

+---------------------------+
|avg(fractal_dimension_mean)|
+---------------------------+
|        0.06286739495798319|
+---------------------------+
```

Figure A.

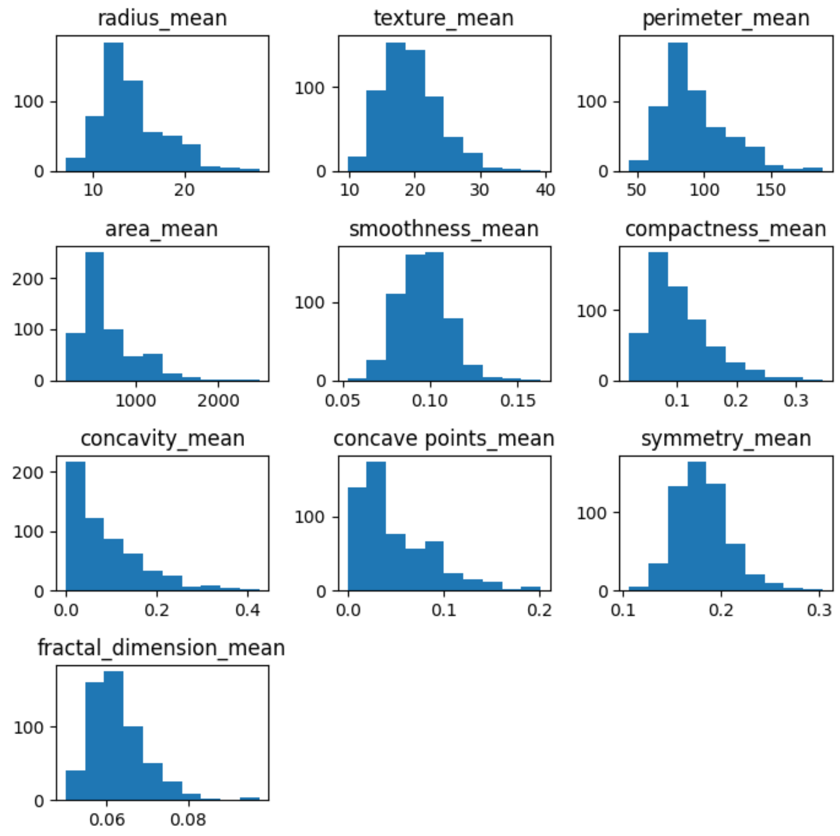


Figure B.

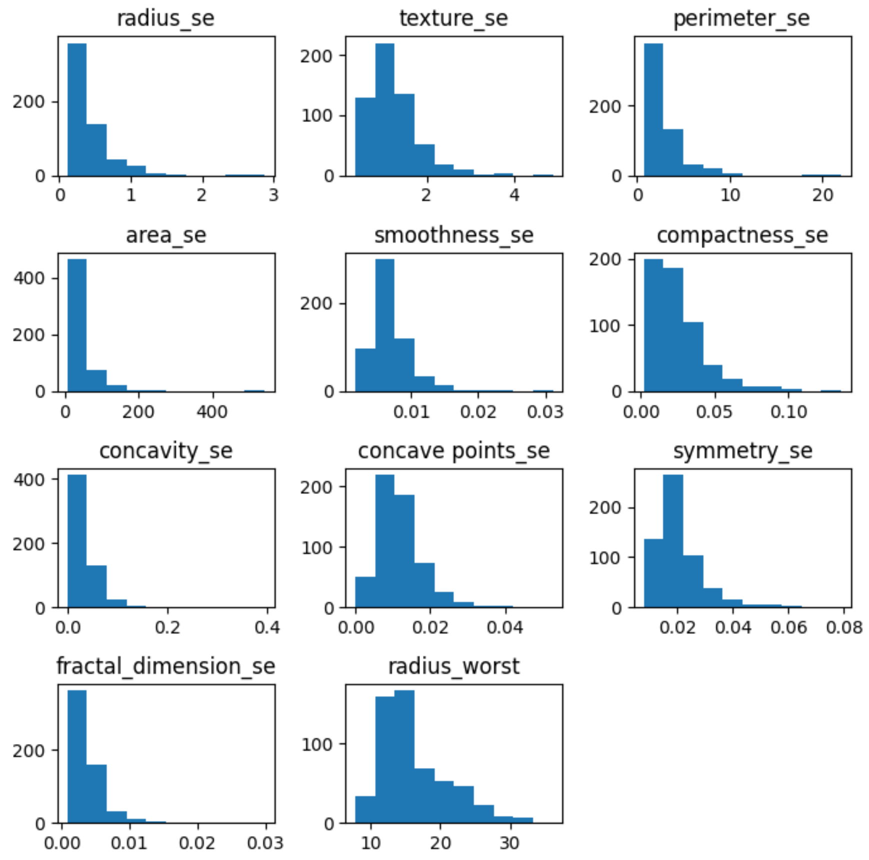


Figure C.

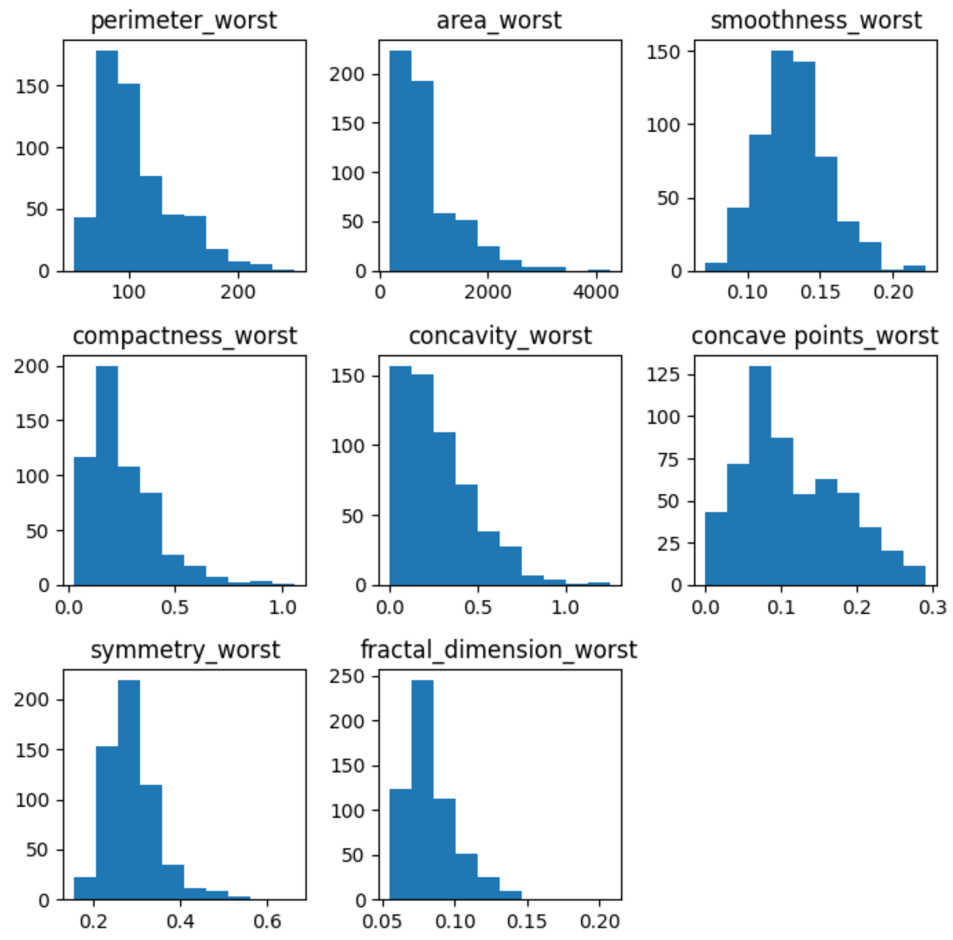
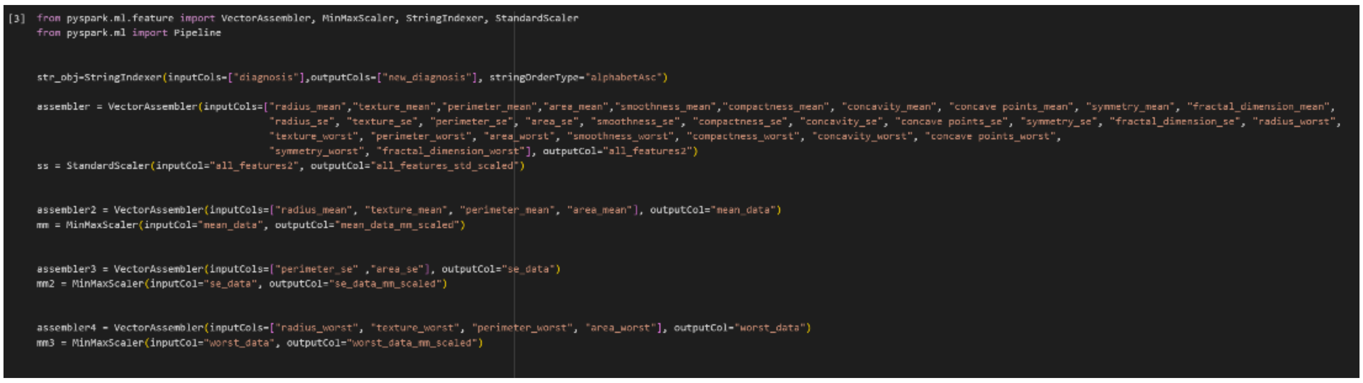
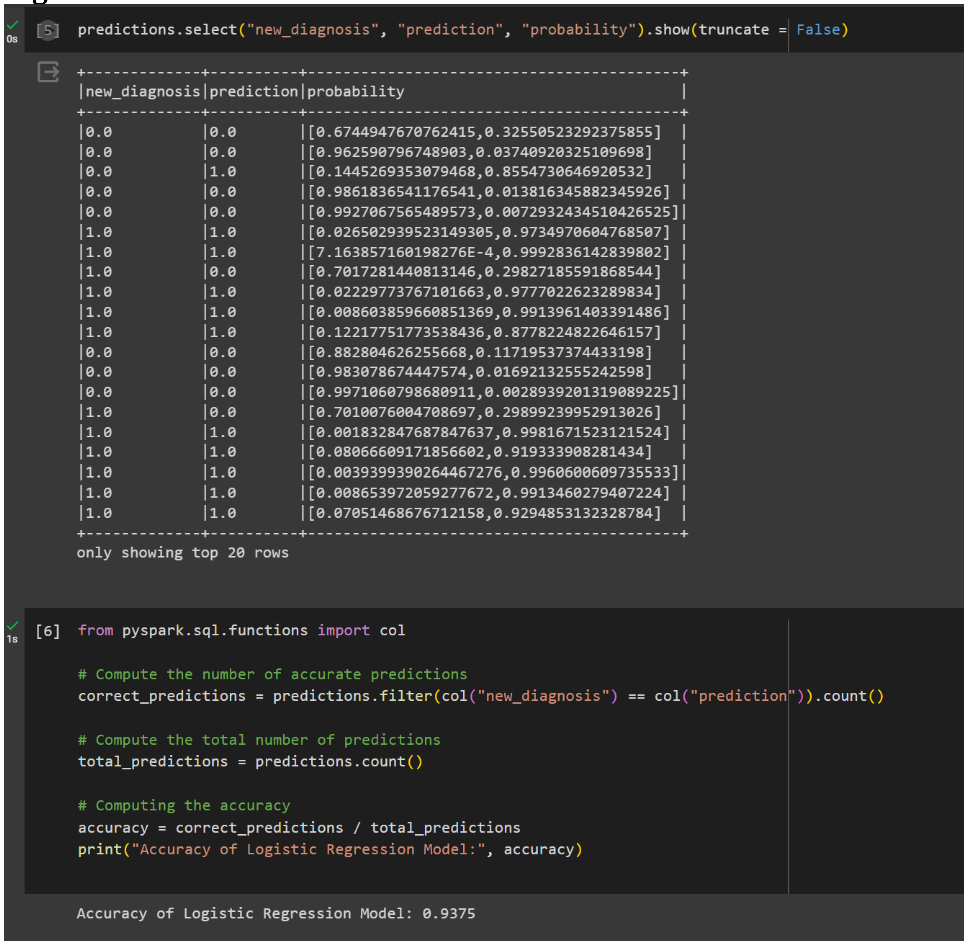


Figure D.



```
[3]  from pyspark.ml.feature import VectorAssembler, MinMaxScaler, StringIndexer, StandardScaler
     from pyspark.ml import Pipeline


     str_obj=StringIndexer(inputCols=["diagnosis"],outputCols=["new_diagnosis"], stringOrderType="alphabetAsc")

     assembler = VectorAssembler(inputCols=["radius_mean","texture_mean","perimeter_mean","area_mean","smoothness_mean","compactness_mean", "concavity_mean", "concave points_mean", "symmetry_mean", "fractal_dimension_mean",
                                            "radius_se", "texture_se", "perimeter_se", "area_se", "smoothness_se", "compactness_se", "concavity_se", "concave points_se", "symmetry_se", "fractal_dimension_se", "radius_worst",
                                            "texture_worst", "perimeter_worst", "area_worst", "smoothness_worst", "compactness_worst", "concavity_worst", "concave points_worst",
                                            "symmetry_worst", "fractal_dimension_worst"], outputCol="all_features2")
     ss = StandardScaler(inputCol="all_features2", outputCol="all_features_std_scaled")


     assembler2 = VectorAssembler(inputCols=["radius_mean", "texture_mean", "perimeter_mean", "area_mean"], outputCol="mean_data")
     mm = MinMaxScaler(inputCol="mean_data", outputCol="mean_data_mm_scaled")


     assembler3 = VectorAssembler(inputCols=["perimeter_se" ,"area_se"], outputCol="se_data")
     mm2 = MinMaxScaler(inputCol="se_data", outputCol="se_data_mm_scaled")


     assembler4 = VectorAssembler(inputCols=["radius_worst", "texture_worst", "perimeter_worst", "area_worst"], outputCol="worst_data")
     mm3 = MinMaxScaler(inputCol="worst_data", outputCol="worst_data_mm_scaled")
```

Figure E.



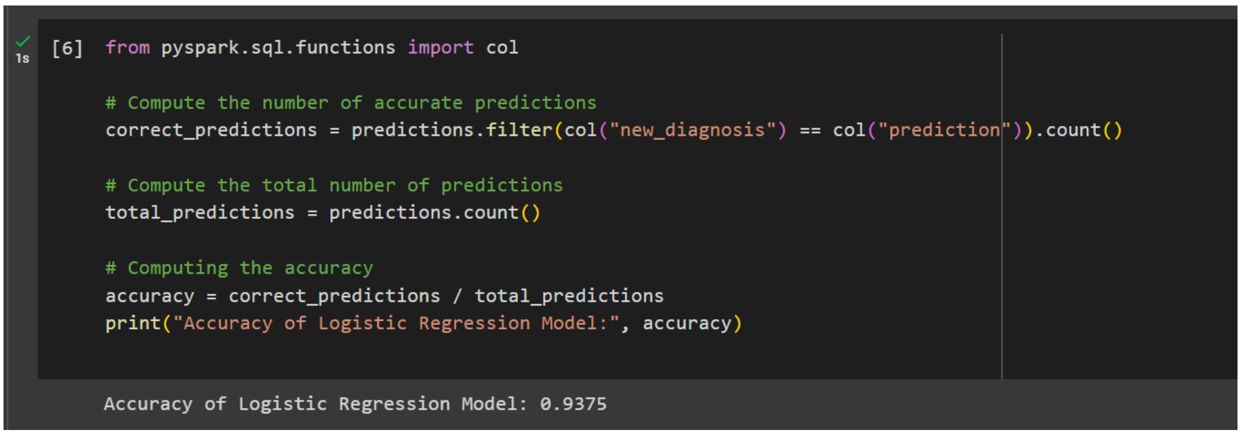
```
[5] predictions.select("new_diagnosis", "prediction", "probability").show(truncate = False)
```

```
+-------------+----------+------------------------------------------+
|new_diagnosis|prediction|probability                               |
+-------------+----------+------------------------------------------+
|0.0          |0.0       |[0.6744947670762415,0.32550523292375855]  |
|0.0          |0.0       |[0.962590796748903,0.03740920325109698]   |
|0.0          |1.0       |[0.1445269353079468,0.8554730646920532]   |
|0.0          |0.0       |[0.9861836541176541,0.013816345882345926] |
|0.0          |0.0       |[0.9927067565489573,0.0072932434510426525]|
|1.0          |1.0       |[0.026502939523149305,0.9734970604768507] |
|1.0          |1.0       |[7.163857160198276E-4,0.9992836142839802] |
|1.0          |0.0       |[0.7017281440813146,0.29827185591868544]  |
|1.0          |1.0       |[0.02229773767101663,0.9777022623289834]  |
|1.0          |1.0       |[0.008603859660851369,0.9913961403391486] |
|1.0          |1.0       |[0.12217751773538436,0.8778224822646157]  |
|0.0          |0.0       |[0.882804626255668,0.11719537374433198]   |
|0.0          |0.0       |[0.983078674447574,0.01692132555242598]   |
|0.0          |0.0       |[0.9971060798680911,0.0028939201319089225]|
|1.0          |0.0       |[0.7010076004708697,0.29899239952913026]  |
|1.0          |1.0       |[0.001832847687847637,0.9981671523121524] |
|1.0          |1.0       |[0.08066609171856602,0.919333908281434]   |
|1.0          |1.0       |[0.0039399390264467276,0.9960600609735533]|
|1.0          |1.0       |[0.008653972059277672,0.9913460279407224] |
|1.0          |1.0       |[0.07051468676712158,0.9294853132328784]  |
+-------------+----------+------------------------------------------+
only showing top 20 rows
```

```
[6] from pyspark.sql.functions import col

    # Compute the number of accurate predictions
    correct_predictions = predictions.filter(col("new_diagnosis") == col("prediction")).count()

    # Compute the total number of predictions
    total_predictions = predictions.count()

    # Computing the accuracy
    accuracy = correct_predictions / total_predictions
    print("Accuracy of Logistic Regression Model:", accuracy)
```

```
Accuracy of Logistic Regression Model: 0.9375
```

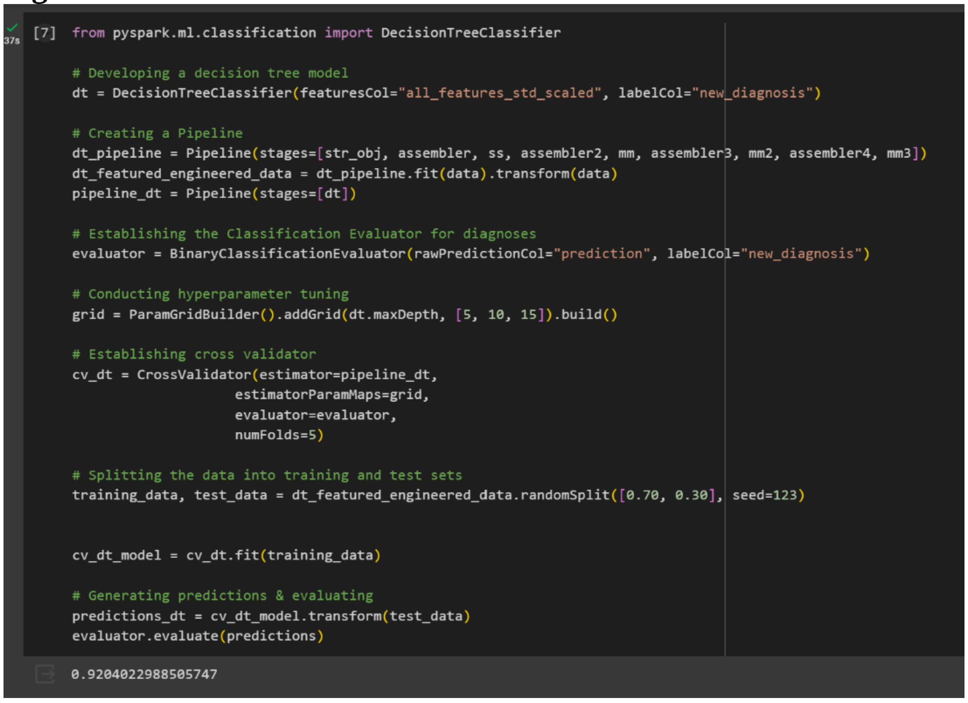
Figure F.



```
[6] from pyspark.sql.functions import col

    # Compute the number of accurate predictions
    correct_predictions = predictions.filter(col("new_diagnosis") == col("prediction")).count()

    # Compute the total number of predictions
    total_predictions = predictions.count()

    # Computing the accuracy
    accuracy = correct_predictions / total_predictions
    print("Accuracy of Logistic Regression Model:", accuracy)
```

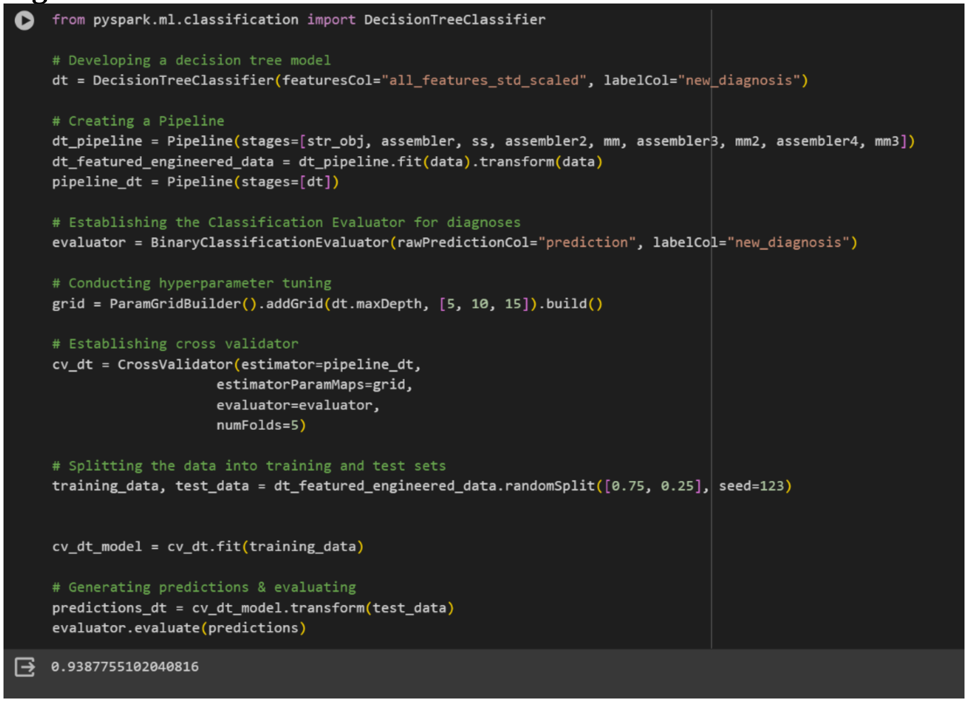
```
Accuracy of Logistic Regression Model: 0.9375
```

Figure G.



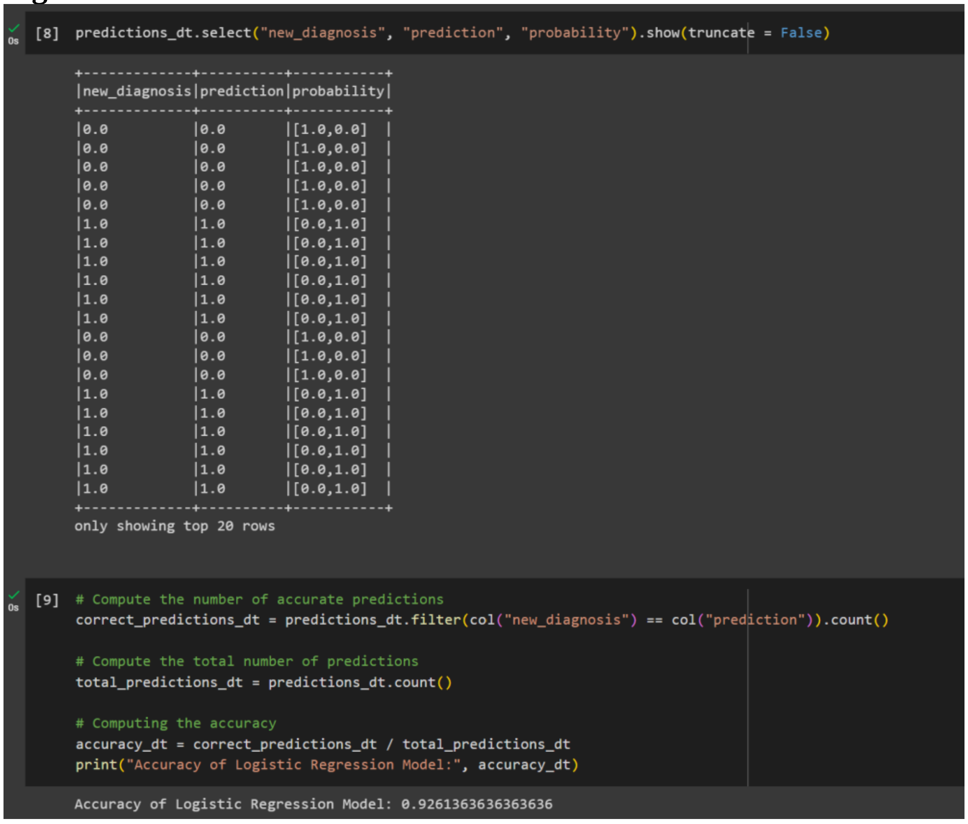
```
[7] from pyspark.ml.classification import DecisionTreeClassifier

    # Developing a decision tree model
    dt = DecisionTreeClassifier(featuresCol="all_features_std_scaled", labelCol="new_diagnosis")

    # Creating a Pipeline
    dt_pipeline = Pipeline(stages=[str_obj, assembler, ss, assembler2, mm, assembler3, mm2, assembler4, mm3])
    dt_featured_engineered_data = dt_pipeline.fit(data).transform(data)
    pipeline_dt = Pipeline(stages=[dt])

    # Establishing the Classification Evaluator for diagnoses
    evaluator = BinaryClassificationEvaluator(rawPredictionCol="prediction", labelCol="new_diagnosis")

    # Conducting hyperparameter tuning
    grid = ParamGridBuilder().addGrid(dt.maxDepth, [5, 10, 15]).build()

    # Establishing cross validator
    cv_dt = CrossValidator(estimator=pipeline_dt,
                        estimatorParamMaps=grid,
                        evaluator=evaluator,
                        numFolds=5)

    # Splitting the data into training and test sets
    training_data, test_data = dt_featured_engineered_data.randomSplit([0.70, 0.30], seed=123)


    cv_dt_model = cv_dt.fit(training_data)

    # Generating predictions & evaluating
    predictions_dt = cv_dt_model.transform(test_data)
    evaluator.evaluate(predictions)

    0.9204022988505747
```

Figure H.



```
from pyspark.ml.classification import DecisionTreeClassifier

# Developing a decision tree model
dt = DecisionTreeClassifier(featuresCol="all_features_std_scaled", labelCol="new_diagnosis")

# Creating a Pipeline
dt_pipeline = Pipeline(stages=[str_obj, assembler, ss, assembler2, mm, assembler3, mm2, assembler4, mm3])
dt_featured_engineered_data = dt_pipeline.fit(data).transform(data)
pipeline_dt = Pipeline(stages=[dt])

# Establishing the Classification Evaluator for diagnoses
evaluator = BinaryClassificationEvaluator(rawPredictionCol="prediction", labelCol="new_diagnosis")

# Conducting hyperparameter tuning
grid = ParamGridBuilder().addGrid(dt.maxDepth, [5, 10, 15]).build()

# Establishing cross validator
cv_dt = CrossValidator(estimator=pipeline_dt,
                    estimatorParamMaps=grid,
                    evaluator=evaluator,
                    numFolds=5)

# Splitting the data into training and test sets
training_data, test_data = dt_featured_engineered_data.randomSplit([0.75, 0.25], seed=123)


cv_dt_model = cv_dt.fit(training_data)

# Generating predictions & evaluating
predictions_dt = cv_dt_model.transform(test_data)
evaluator.evaluate(predictions)

0.9387755102040816
```

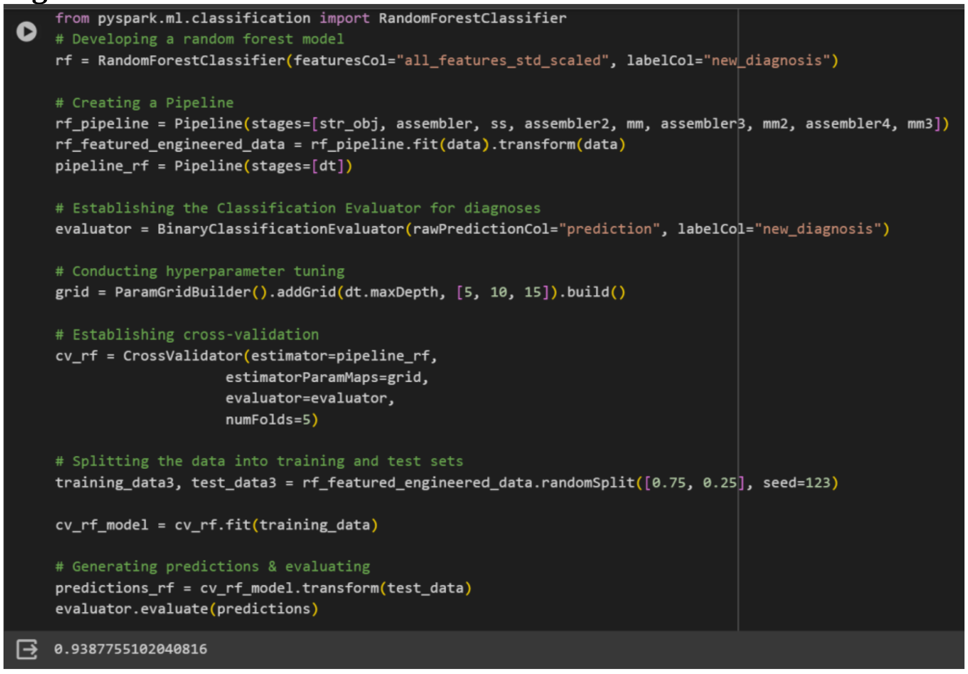
Figure I.



```
[8] predictions_dt.select("new_diagnosis", "prediction", "probability").show(truncate = False)

+-------------+----------+-----------+
|new_diagnosis|prediction|probability|
+-------------+----------+-----------+
|0.0          |0.0       |[1.0,0.0]  |
|0.0          |0.0       |[1.0,0.0]  |
|0.0          |0.0       |[1.0,0.0]  |
|0.0          |0.0       |[1.0,0.0]  |
|0.0          |0.0       |[1.0,0.0]  |
|1.0          |1.0       |[0.0,1.0]  |
|1.0          |1.0       |[0.0,1.0]  |
|1.0          |1.0       |[0.0,1.0]  |
|1.0          |1.0       |[0.0,1.0]  |
|1.0          |1.0       |[0.0,1.0]  |
|1.0          |1.0       |[0.0,1.0]  |
|0.0          |0.0       |[1.0,0.0]  |
|0.0          |0.0       |[1.0,0.0]  |
|0.0          |0.0       |[1.0,0.0]  |
|1.0          |1.0       |[0.0,1.0]  |
|1.0          |1.0       |[0.0,1.0]  |
|1.0          |1.0       |[0.0,1.0]  |
|1.0          |1.0       |[0.0,1.0]  |
|1.0          |1.0       |[0.0,1.0]  |
|1.0          |1.0       |[0.0,1.0]  |
+-------------+----------+-----------+
only showing top 20 rows
```

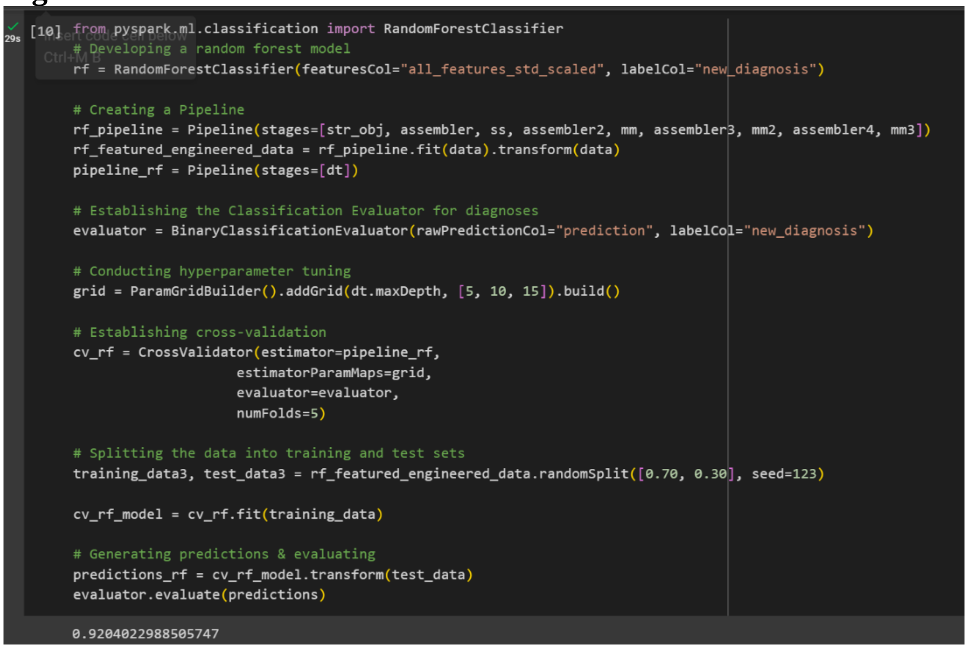
```
[9] # Compute the number of accurate predictions
    correct_predictions_dt = predictions_dt.filter(col("new_diagnosis") == col("prediction")).count()

    # Compute the total number of predictions
    total_predictions_dt = predictions_dt.count()

    # Computing the accuracy
    accuracy_dt = correct_predictions_dt / total_predictions_dt
    print("Accuracy of Logistic Regression Model:", accuracy_dt)

Accuracy of Logistic Regression Model: 0.9261363636363636
```

Figure J.



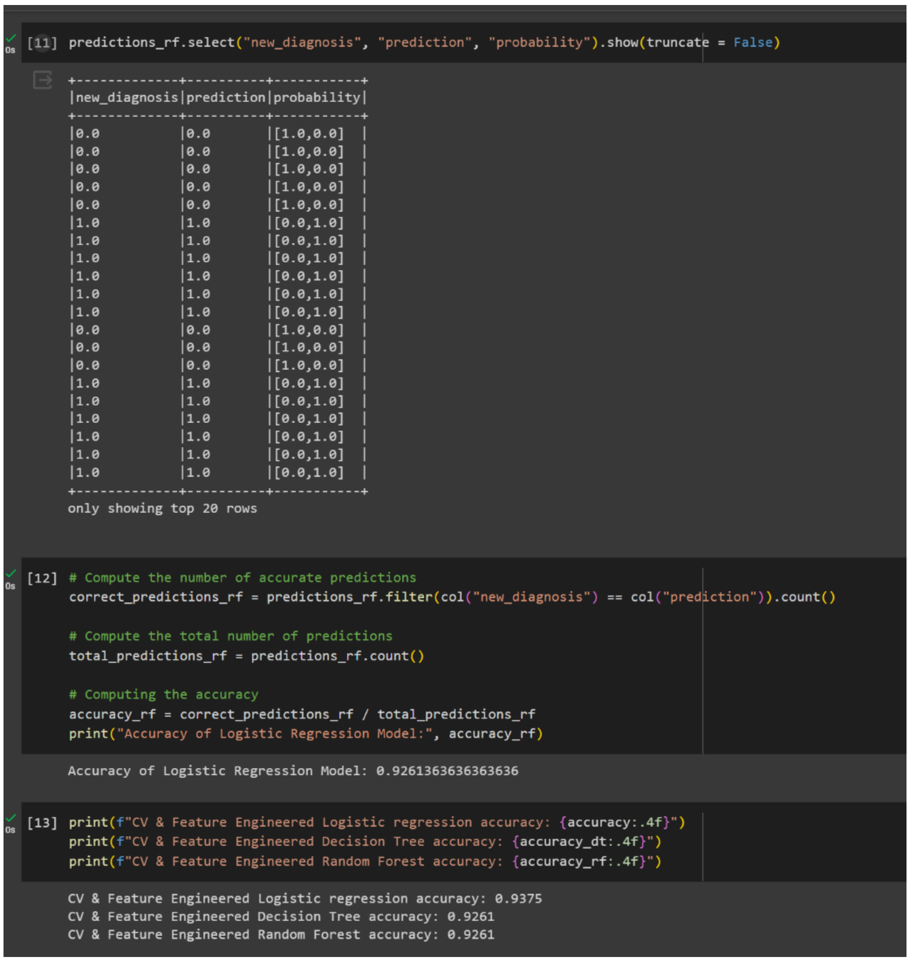
```
from pyspark.ml.classification import RandomForestClassifier
# Developing a random forest model
rf = RandomForestClassifier(featuresCol="all_features_std_scaled", labelCol="new_diagnosis")

# Creating a Pipeline
rf_pipeline = Pipeline(stages=[str_obj, assembler, ss, assembler2, mm, assembler3, mm2, assembler4, mm3])
rf_featured_engineered_data = rf_pipeline.fit(data).transform(data)
pipeline_rf = Pipeline(stages=[dt])

# Establishing the Classification Evaluator for diagnoses
evaluator = BinaryClassificationEvaluator(rawPredictionCol="prediction", labelCol="new_diagnosis")

# Conducting hyperparameter tuning
grid = ParamGridBuilder().addGrid(dt.maxDepth, [5, 10, 15]).build()

# Establishing cross-validation
cv_rf = CrossValidator(estimator=pipeline_rf,
                       estimatorParamMaps=grid,
                       evaluator=evaluator,
                       numFolds=5)

# Splitting the data into training and test sets
training_data3, test_data3 = rf_featured_engineered_data.randomSplit([0.75, 0.25], seed=123)

cv_rf_model = cv_rf.fit(training_data)

# Generating predictions & evaluating
predictions_rf = cv_rf_model.transform(test_data)
evaluator.evaluate(predictions)

0.9387755102040816
```

Figure K.



```
[10] from pyspark.ml.classification import RandomForestClassifier
     # Developing a random forest model
     rf = RandomForestClassifier(featuresCol="all_features_std_scaled", labelCol="new_diagnosis")

     # Creating a Pipeline
     rf_pipeline = Pipeline(stages=[str_obj, assembler, ss, assembler2, mm, assembler3, mm2, assembler4, mm3])
     rf_featured_engineered_data = rf_pipeline.fit(data).transform(data)
     pipeline_rf = Pipeline(stages=[dt])

     # Establishing the Classification Evaluator for diagnoses
     evaluator = BinaryClassificationEvaluator(rawPredictionCol="prediction", labelCol="new_diagnosis")

     # Conducting hyperparameter tuning
     grid = ParamGridBuilder().addGrid(dt.maxDepth, [5, 10, 15]).build()

     # Establishing cross-validation
     cv_rf = CrossValidator(estimator=pipeline_rf,
                         estimatorParamMaps=grid,
                         evaluator=evaluator,
                         numFolds=5)

     # Splitting the data into training and test sets
     training_data3, test_data3 = rf_featured_engineered_data.randomSplit([0.70, 0.30], seed=123)

     cv_rf_model = cv_rf.fit(training_data)

     # Generating predictions & evaluating
     predictions_rf = cv_rf_model.transform(test_data)
     evaluator.evaluate(predictions)

     0.9204022988505747
```

Figure L.



```
[11] predictions_rf.select("new_diagnosis", "prediction", "probability").show(truncate = False)

     +-------------+----------+-----------+
     |new_diagnosis|prediction|probability|
     +-------------+----------+-----------+
     |0.0          |0.0       |[1.0,0.0]  |
     |0.0          |0.0       |[1.0,0.0]  |
     |0.0          |0.0       |[1.0,0.0]  |
     |0.0          |0.0       |[1.0,0.0]  |
     |0.0          |0.0       |[1.0,0.0]  |
     |1.0          |1.0       |[0.0,1.0]  |
     |1.0          |1.0       |[0.0,1.0]  |
     |1.0          |1.0       |[0.0,1.0]  |
     |1.0          |1.0       |[0.0,1.0]  |
     |1.0          |1.0       |[0.0,1.0]  |
     |1.0          |1.0       |[0.0,1.0]  |
     |0.0          |0.0       |[1.0,0.0]  |
     |0.0          |0.0       |[1.0,0.0]  |
     |0.0          |0.0       |[1.0,0.0]  |
     |1.0          |1.0       |[0.0,1.0]  |
     |1.0          |1.0       |[0.0,1.0]  |
     |1.0          |1.0       |[0.0,1.0]  |
     |1.0          |1.0       |[0.0,1.0]  |
     |1.0          |1.0       |[0.0,1.0]  |
     |1.0          |1.0       |[0.0,1.0]  |
     +-------------+----------+-----------+
     only showing top 20 rows

[12] # Compute the number of accurate predictions
     correct_predictions_rf = predictions_rf.filter(col("new_diagnosis") == col("prediction")).count()

     # Compute the total number of predictions
     total_predictions_rf = predictions_rf.count()

     # Computing the accuracy
     accuracy_rf = correct_predictions_rf / total_predictions_rf
     print("Accuracy of Logistic Regression Model:", accuracy_rf)

     Accuracy of Logistic Regression Model: 0.9261363636363636

[13] print(f"CV & Feature Engineered Logistic regression accuracy: {accuracy:.4f}")
     print(f"CV & Feature Engineered Decision Tree accuracy: {accuracy_dt:.4f}")
     print(f"CV & Feature Engineered Random Forest accuracy: {accuracy_rf:.4f}")

     CV & Feature Engineered Logistic regression accuracy: 0.9375
     CV & Feature Engineered Decision Tree accuracy: 0.9261
     CV & Feature Engineered Random Forest accuracy: 0.9261
```

References

https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic
https://www.cdc.gov/cancer/breast/basic_info/index.htm#:~:text=Each%20year%20in%20the%20United,cancer%20than%20all%20other%20women.