MANCHESTER
1824

Professor Mark Muldoon | Professor Diego Perez Ruiz
DATA 70121 | Assignment Report

Coursework: EDA & Regression

Student ID: 11493842
MSc Data Science
The University of Manchester

DATA 70121 | Statistics and Machine Learning: 1
20TH November 2023

1.1 Summary
The PimaDiabetes dataset used for the analysis and preparation of this report originates from the National Institute of Diabetes and Digestive and Kidney Diseases in the USA. The analyses performed suggested that women with fewer than 7 pregnancies have a 28.5% chance of getting diabetes, while women with 7 or more pregnancies stood at 56.5%. Furthermore, machine learning models used for predictive analytics achieved an accuracy of 83.18% for this dataset.

1.2 Data Description
The PimaDiabetes dataset used for the study contained measurements from 750 women along with the ‘Outcome’ binary variable (0/1) indicating the Diabetes diagnosis. The table provides potential relations between various health indicators and diabetes risk in women. The dataset has 9 attributes which are described as follows:

- Pregnancies - Number of times a woman has been pregnant
- Glucose - Plasma glucose conc. in mg/dl, 2 hours post Oral Glucose Tolerance(OGT) Test
- Blood_Pressure - Diastolic blood pressure in mm Hg.
- Skin_Thickness - Tricep skin fold thickness in mm.
- Serum Insulin - Insulin concentration in µU/ml 2 hours post OGT test.
- BMI (Body Mass Index) - Individual's weight in kg divided by their height in $(\text{meters})^2$
- Diabetes Pedigree - Number suggesting diabetes risk due to the genetic influence of the woman’s diabetic and non-diabetic relatives. Higher scores mean more close relatives with diabetes
- Age - Age, measured in years
- Outcome - Binary variable with '1' as a positive diagnosis of diabetes, and '0' indicating no diabetes.

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1

Fig 1. Sample snippet of the PimaDiabetes Dataset

1.3 Data and Unit Errors

- All the entries in the dataset are not medically plausible. Some of the entries especially in the attributes of Blood Pressure, BMI, Glucose and Insulin are 0. This suggests inconsistency in data handling on the part of surveyors. It might be the case that the null values during data collection were replaced by 0. These discrepancies have affected the results of the analysis.
- Secondly, the unit for Insulin levels has not been mentioned clearly. The values that appear in the dataset do not appear to be written in the international unit system i.e. (μU/ml), but rather in picomoles per litre.
- Thirdly, all the attributes contain extreme outlier values that might be a result of human error during collection. The values however are not large in number.

2. Exploratory Data Analysis, EDA

The following table lists basic statistical values namely the number of rows, mean, standard deviation, minimum, maximum, and the [25th, 50th, 75th] percentile values for all of the attributes. The missing and outlier values have skewed the distributions.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age	Outcome
count	750.000000	750.000000	750.000000	750.000000	750.000000	750.000000	750.000000	750.000000	750.000000
mean	3.844000	120.737333	68.982667	20.489333	80.378667	31.959067	0.473544	33.166667	0.346667
std	3.370085	32.019671	19.508814	15.918828	115.019198	7.927399	0.332119	11.708872	0.476226
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.244000	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	36.500000	32.000000	0.377000	29.000000	0.000000
75%	6.000000	140.750000	80.000000	32.000000	129.750000	36.575000	0.628500	40.750000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

Fig 2. Basic statistical measures and tendencies for the 9 attributes

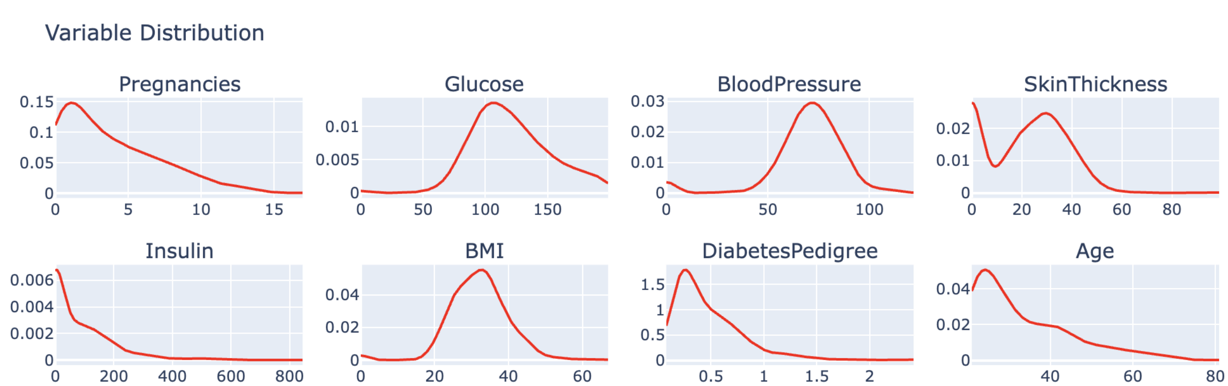

Fig 3. Distributions for the attributes

The kernel density estimate plots are skewed because of the errors that were discussed earlier. All of the attributes have outlier values as well as missing values.

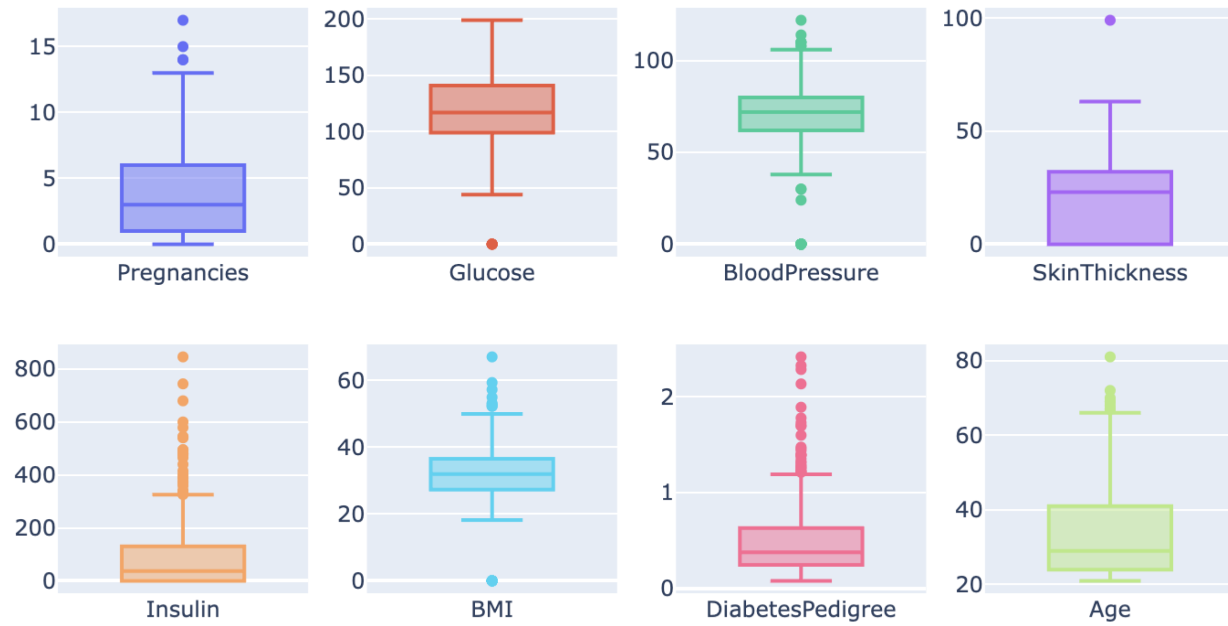

Fig 4: Boxplot distributions for Outliers and Quartile distributions

Fig 4. The 25th, 50th and 75th percentiles of the attributes are denoted by the 3 lines of the boxplot, while the maximum and minimum values lie at the extremes.

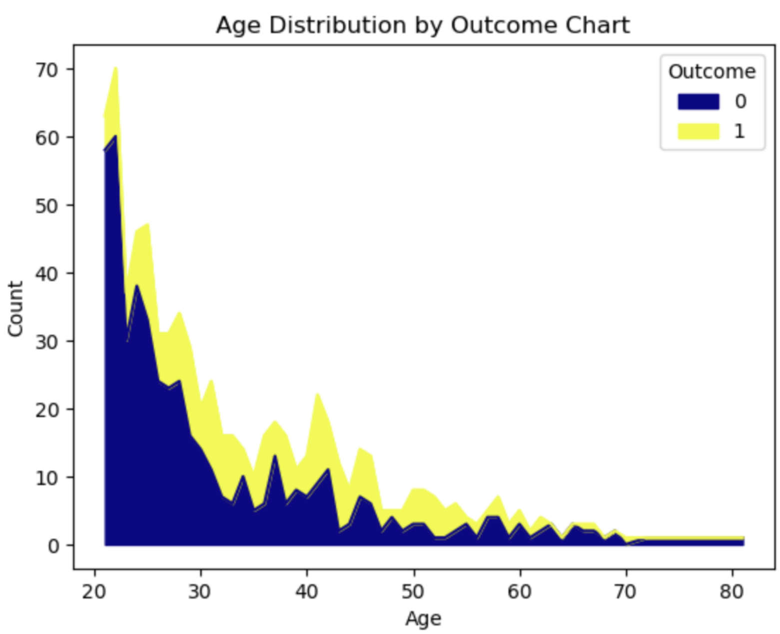

Fig. 5 represents the age distribution by diabetes outcome values. We can infer that the majority of the population that gets pregnant between the ages 20 and 30, has a very low chance of getting diabetes. However, as the age increases the chance of getting diabetes increases.

The correlation matrix outputs the correlation coefficients (-1 ≤ r ≤ 1) between the attributes, the values of which were used to carry out further EDA.

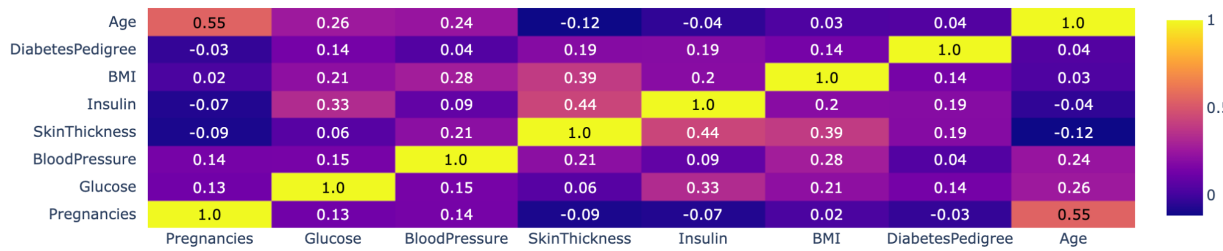

Fig 6. Correlation matrix (8 x 8)

The scatter plot in Fig. 7 denotes the relation between Glucose and Blood Pressure in relation to diabetes outcomes. From the graph, we can see a visible cluster of red dots, which denotes that higher glucose levels paired with high blood pressure increase the chance of having diabetes.

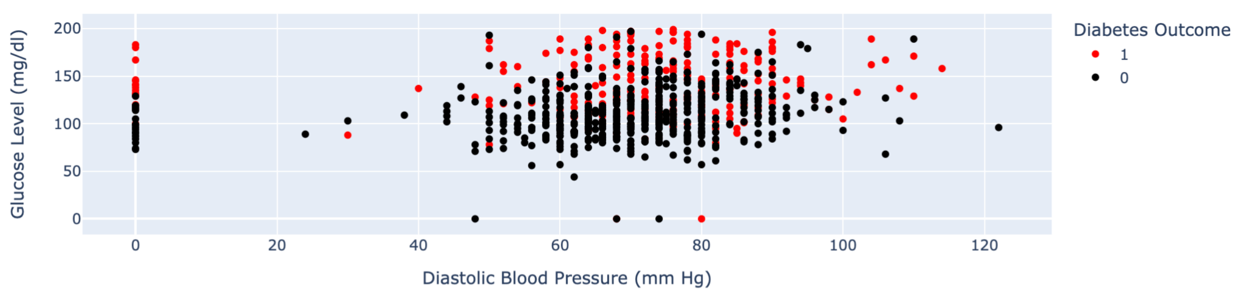

Fig. 7 Blood Pressure vs Glucose by Diabetes Outcome

The data used in Fig.8 was divided into 3 age groups such that all of the groups have almost the same number of data points. For ages 25 and less the probability of getting diabetes is 16.7%. For the group with ages between 25 and 36, the probability is 39.4%. For ages between 36 and 81, the probability is 49.1%.

Fig. 8 Diabetes Outcome with Age Sections

In Fig. 9 the boxplots for the diastolic blood pressures were divided on the basis of binary diabetes outcome variables. Both the plots come out to be almost similar. Hence it can be safely concluded that there is a neutral relation solely between diastolic blood pressure and diabetes outcome.

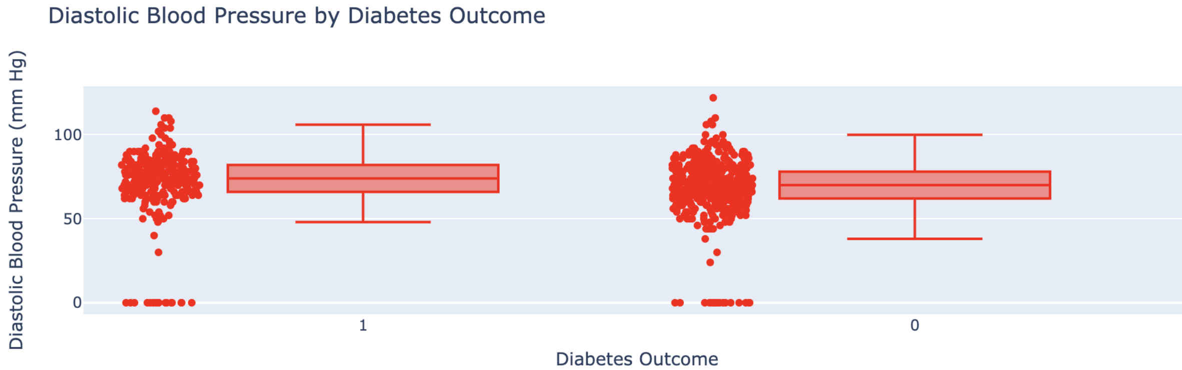

Fig. 9 Diastolic Blood Pressure by Diabetes Outcome

The data used in Fig.10 was divided into 3 tricep skin thickness groups such that all of the groups have almost the same number of data points. For tricep skin thicknesses less than 24mm the probability of developing diabetes rests at 18.1%. For thickness between 24 and 32mm, the probability lies at 34.7%, while the probability for the range 32mm - 99mm is 46.9%.

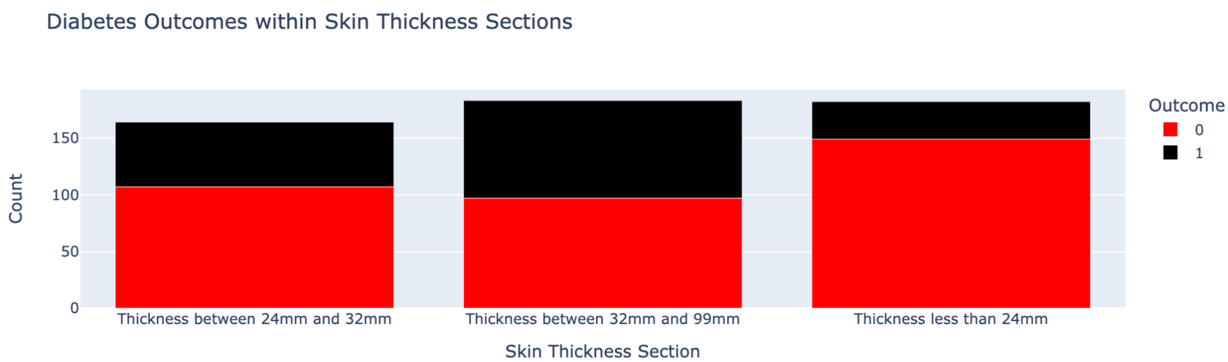

Fig 10 Skin thickness range affecting Diabetes outcome

3. Problem Statement 3

The following is the result output of problem statement 3.
The probability of developing diabetes given that there are:

- 6 or fewer pregnancies is 28.49%
- 7 or more pregnancies is 56.51%

```
Random Forest Training Score 68.26666666666667%

DIABETES PROBABLITY GIVEN THAT SIX OR FEWER PREGNANCIES [not developing, developing]
[0.71507748 0.28492252]
[0.71507748 0.28492252]
...
[0.71507748 0.28492252]
[0.71507748 0.28492252]

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

DIABETES PROBABLITY GIVEN THAT SEVEN OR MORE PREGNANCIES [not developing, developing]
[0.43485087 0.56514913]
[0.43485087 0.56514913]
...
[0.43485087 0.56514913]
[0.43485087 0.56514913]
```

Fig. 11 Results for the Problem Statement 3

4. Problem Statement 4

```
Random Forest Traing Score: 100.0%
Random Forest Accuracy Score: 84.070796460177%


              precision    recall  f1-score   support

           0       0.88      0.88      0.88        73
           1       0.78      0.78      0.78        40

    accuracy                           0.84       113
   macro avg       0.83      0.83      0.83       113
weighted avg       0.84      0.84      0.84       113
```

A random forest classifier model was used in the training, which had a testing accuracy of 84.07% and a training score of 100%. The independent variables chosen for the model were all of the 9 attributes as well as the newer added column called ‘SevenOrMorePregnancies’.

The total number of estimators used in this model was 150. Moreover, the f1_scores for both outcomes are listed in the figure above.
The reason for choosing the RFC model was based on its high accuracy and versatility. The model successfully handles missing values and outliers, both of which are quite prominent in this

dataset. Moreover, normalisation and standardisation of the data were not required as the RFC model does not require scaling.

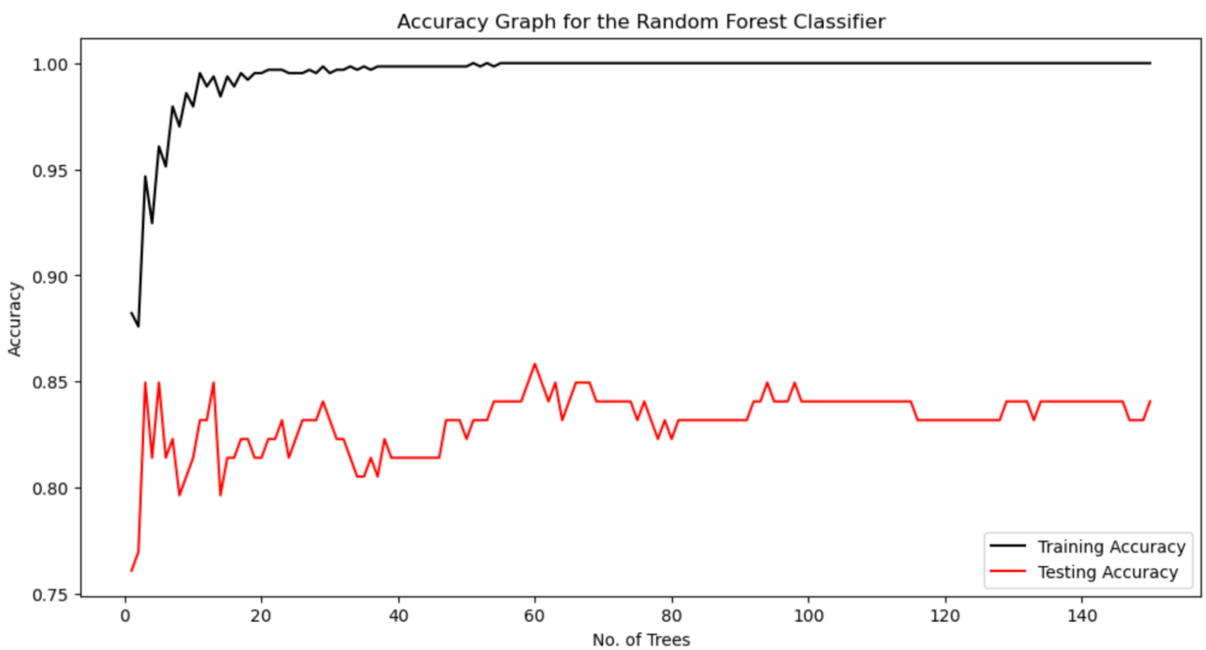

Fig 11. Test accuracy for the RFC training

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age	SevenOrMorePregnancies	Predicted Outcome
4	136	70	0	0	31.2	1.182	22	0	1
1	121	78	39	74	39.0	0.261	28	0	0
3	108	62	24	0	26.0	0.223	25	0	0
0	181	88	44	510	43.3	0.222	26	0	1
8	154	78	32	0	32.4	0.443	45	1	1

Fig 12. Predicted Outcome comes out to be as follows(last column)

References

- Smith, J. W., Everhart, J., Dickson, W., Knowler, W., & Johannes, R. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In Proceedings of the annual symposium on computer application in medical care (pp. 261–265).
- Knopp, J. L., Holder-Pearson, L., & Chase, J. G. (2018, 2023/01/16). Insulin units and conversion factors: A story of truth, boots, and faster half-truths. Journal of Diabetes Science and Technology, 13(3), 597–600. doi: 10.1177/1932296818805074.
- Melmed, S., Polonsky, K. S., Larsen, P. R., & Kronenberg, H. M. (Eds.). (2016). Williams textbook of endocrinology (Thirteenth Edition ed.). Philadelphia: Elsevier. doi: 10.1016/C2013-0-15980- 6.
- https://machinelearningmastery.com/random-forest-ensemble-in-python/

5. Code Snippets

```
#libraries
import matplotlib.pyplot as plt
import plotly.express as px  #Plotly Express for interactive visualizations
import plotly.graph_objects as go  #Graph Objects for Visualisations
import plotly.figure_factory as ff  #Figure Factory for complex visualizations
import seaborn as sns  #Seaborn for statistical visualization
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier# Random Forest classifier
from sklearn.model_selection import train_test_split# Train-test split for model evaluation
from sklearn.metrics import (
   accuracy_score,
   confusion_matrix,  # Confusion matrix for classification evaluation
   classification_report,
   f1_score  # F1 score for model evaluation
)
from plotly.subplots import make_subplots  # Multiple plotting
import plotly
import warnings
warnings.filterwarnings('ignore')
```

FIGURE 2

```
dataset = pd.read_csv("PimaDiabetes.csv") #reading the csv file
# Generating a summary of descriptive statistics for the dataset
dataset.describe(include='all')
```

FIGURE 3

```
# Columns for the distribution plots
col = ["Pregnancies", "Glucose", "BloodPressure", "SkinThickness",
                  "Insulin", "BMI", "DiabetesPedigree", "Age"]
# Creating a subplot with 2 rows and 4 columns
figure = make_subplots(rows=2, cols=4, subplot_titles = col)
# Iteration through each column
for i, column in enumerate(col):
   datahist = [dataset[column].dropna()]
   label = [column]
   fig = ff.create_distplot(datahist, label, show_hist=False, show_rug=False,
curve_type='kde')
   line = fig['data'][0]
   line.update(line=dict(color='red'))
   figure.add_trace(line, row=(i // 4) + 1, col=(i % 4) + 1)
figure.update_layout(title_text='Variable Distribution')
figure.update_layout(margin=dict(t=90, l=60, r=60, b=60))
```

```
figure.update_layout(showlegend=False)
figure.show()
```

FIGURE 4

```
# Convert 'Outcome' column to string type
dataset['Outcome'] = dataset['Outcome'].astype(str)
# Columns for box plots
col = ["Pregnancies", "Glucose", "BloodPressure", "SkinThickness",
                  "Insulin", "BMI", "DiabetesPedigree", "Age"]
# Subplot with 2 rows and 4 columns
figure = make_subplots(rows=2, cols=4)
row_no = 1
col_no = 1
#Iteration through each column
for i, column in enumerate(col):
   figure.add_trace(
       go.Box(y=dataset[column], name=column),
       row=row_no, col=col_no
   )
   col_no += 1 #increment
   if col_no > 4:
       col_no = 1
       row_no += 1
#Subplot
figure.update_layout(height=500, width=900, title_text="Quartiles and Outliers")
figure.show()
```

FIGURE 5

```
# Grouping the Data
age_dist = df.groupby(['Age', 'Outcome']).size().unstack().fillna(0)
# Creating an area plot
age_dist.plot(kind='area', stacked=True, cmap='plasma')
plt.title('Age Distribution by Outcome Chart')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```

FIGURE 6

```
# Calculating the correlation matrix
correlation = dataset.corr()
# Creating the heatmap
figure = ff.create_annotated_heatmap(
   x=list(correlation.columns),
```

```
    y=list(correlation.index),
    z=correlation.values,
    annotation_text=correlation.round(2).values,
    showscale=True,
    colorscale='plasma'
)
#Figure
figure.update_xaxes(side="bottom")
figure.update_layout(title_text='Correlation Matrix for all Variables')
figure.show()
```

FIGURE 7

```
# Creating a scatter plot
figure = px.scatter(dataset, x='BloodPressure', y='Glucose', color='Outcome',
                labels={
                    "Outcome": "Diabetes Outcome",
                    "BloodPressure": "Diastolic Blood Pressure (mm Hg) ",
                    "Glucose": "Glucose Level (mg/dl) "
                },
                title="Blood Pressure vs Glucose by Diabetes Outcome",
                color_discrete_map={'0': 'black', '1': 'red'})
# Scatter Plot Display
figure.show()
```

FIGURE 8

```
# Defining a function to categorize age into different sections sections
def age_col(age):
    if age <= 25: return 'Age less than 25'
    elif 25 < age <= 36: return 'Age between 25 and 36'
    elif 36< age <= 81: return 'Age between 36 and 81'
# Creating a new age section column
dataset['Age_Section'] = dataset['Age'].apply(age_col)
# Grouping and Calculating counts
age_outcome_counts = dataset.groupby(['Age_Section',
'Outcome']).size().reset_index(name='Count')
# Bar chart creation and Display
figure = px.bar(age_outcome_counts, x='Age_Section', y='Count', color='Outcome',
            title='Diabetes Outcomes within Age Sections',
            labels={"Age_Section": "Age Section"},
            color_discrete_map={'0': 'red', '1': 'black'})
figure.show()
```

FIGURE 9

```
# Creating a box plot
figure = px.box(dataset, x='Outcome', y='BloodPressure', points="all",
               boxmode='overlay',
               labels={
                   "Outcome": "Diabetes Outcome",
                   "BloodPressure": "Diastolic Blood Pressure (mm Hg)"
               },
               title="Diastolic Blood Pressure by Diabetes Outcome")
figure.update_traces(marker=dict(color='red'), selector=dict(type='box'))
# Display
figure.show()
```

FIGURE 10

```
# Defining a function to categorize skin thickness into different sections
def skinthicckness_col(skin):
   if 1 < skin <= 24: return 'Thickness less than 24mm'
   elif 24 < skin <= 32: return 'Thickness between 24mm and 32mm'
   elif 32 < skin <= 99: return 'Thickness between 32mm and 99mm'
# Creating a new age section column
dataset['Skinthickness_Section'] = dataset['SkinThickness'].apply(skinthicckness_col)
# Grouping and Calculating counts
skin_outcome_counts = dataset.groupby(['Skinthickness_Section',
'Outcome']).size().reset_index(name='Count')
# Bar chart creation and Display
figure = px.bar(skin_outcome_counts, x='Skinthickness_Section', y='Count',
color='Outcome',
            title='Diabetes Outcomes within Skin Thickness Sections',
            labels={"Skinthickness_Section": "Skin Thickness Section"},
            color_discrete_map={'0': 'red', '1': 'black'})
figure.show()
```

Problem Statement 3 and FIGURE 11

```
# Adding new column "SevenOrMorePregnancies"
dataset["SevenOrMorePregnancies"] = ''
# Looping through each row
for i in range(0, len(dataset)):
   if dataset["Pregnancies"][i] >= 7: #Pregnancies greater than or equal to 7
       dataset["SevenOrMorePregnancies"][i] = 1 # Setting this value to 1
   else:
       dataset["SevenOrMorePregnancies"][i] = 0

# Converting this datatype to integers
```

```python
dataset["SevenOrMorePregnancies"] = dataset["SevenOrMorePregnancies"].astype(int)
#Relevant columns for this model
def_col = dataset[["SevenOrMorePregnancies", "Outcome"]]

# Input variable and Target Variable
var1 = def_col[['SevenOrMorePregnancies']]
var2 = def_col['Outcome']
# Creating a RandomForestClassifier with 500 trees and 'entropy' as the criterio
rand_for = RandomForestClassifier(n_estimators = 500, criterion = 'entropy')
# Training the model for given variables
rand_for.fit(var1, var2)
# Outputting the training score
print("Random Forest Training Score {}%".format(rand_for.score(var1, var2)*100))


# Filtering out the rows where the values are 0 and resetting the index
sixandfew = var1[var1["SevenOrMorePregnancies"] == 0]
sixandfew.reset_index(inplace = True, drop = True)
# Filtering out the rows where the values are 1 and resetting the index
sevenandmore = var1[var1["SevenOrMorePregnancies"] == 1]
sevenandmore.reset_index(inplace = True, drop = True)


# Summarizing the statistics of the array
def stat_summary(array, head=2, tail=2):
   for i in range(head):
       print(array[i])
   print("...\n" * (array.ndim - 1), end="")
   for i in range(-tail, 0, 1):
       print(array[i])

# Displaying the predicted probabilities
print(" ")
print("DIABETES PROBABLITY GIVEN THAT SIX OR FEWER PREGNANCIES [not developing,
developing] ")
stat_summary(rand_for.predict_proba(sixandfew))
print(" ")
print("- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  ")
print(" ")
print("DIABETES PROBABLITY GIVEN THAT SEVEN OR MORE PREGNANCIES [not developing,
developing] ")
stat_summary(rand_for.predict_proba(sevenandmore))
```

Problem Statement 4

```
# Reading data from the two csv files, given and to_predict
defcol = pd.read_csv("PimaDiabetes.csv")
topredict = pd.read_csv("ToPredict.csv")

# Addition of the column seven or more pregnancies to the dataset
defcol["SevenOrMorePregnancies"] = ''
# Looping through each row
for i in range(0, len(defcol)):
   if defcol["Pregnancies"][i] >= 7: # Pregnancies greater than or equal to 7
       defcol["SevenOrMorePregnancies"][i] = 1 # Setting the value to 1
   else:
       defcol["SevenOrMorePregnancies"][i] = 0

# Converting the datatype to integers
defcol["SevenOrMorePregnancies"] = defcol["SevenOrMorePregnancies"].astype(int)

correlation = defcol.corr() # Correlation matrix
# Creation of the heatmap
figure = ff.create_annotated_heatmap(
   x=list(correlation.columns),
   y=list(correlation.index),
   z=correlation.values,
   annotation_text=correlation.round(2).values,
   showscale=True,
   colorscale='Plasma'
)
# Displaying the Figure
figure.update_layout(title_text='Correlation Matrix for all variables')
figure.update_xaxes(side="bottom")
figure.show()
```

FIGURE 11

```
#Choosing Independent variables for training and testing
x = defcol[["Pregnancies", "Glucose", "BloodPressure", "SkinThickness", "Insulin",
"BMI", "DiabetesPedigree", "Age", "SevenOrMorePregnancies"]]
y = defcol["Outcome"] # Selection of the target variable

#Training and testing sets
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.15, random_state=0)

# RandomForestClassifier with 150 trees. Random seed setting
```

```
rand_for = RandomForestClassifier(n_estimators=150, random_state=50)

training_accu = [] # Store training and testing accuracies
testing_accu = []

treelist = range(1, rand_for.n_estimators + 1) # Generating a list of tree numbers

# Looping through different number of trees. Evaluation of the model.
for n_trees in treelist:
   rand_for.set_params(n_estimators=n_trees) # Number of trees
   rand_for.fit(xtrain, ytrain) # training

   training_pred = rand_for.predict(xtrain) # Outcome prediction
   testing_pred = rand_for.predict(xtest)

   training_accu.append(accuracy_score(ytrain, training_pred)) # Calculating and
storing accuracy
   testing_accu.append(accuracy_score(ytest, testing_pred))

# Plotting the two accuracies over the number of trees
plt.figure(figsize=(12, 6))
plt.plot(treelist, training_accu, label='Training Accuracy', color='black')
plt.plot(treelist, testing_accu, label='Testing Accuracy', color='red')

# Creating and Displaying the plot
plt.title('Accuracy Graph for the Random Forest Classifier')
plt.xlabel('No. of Trees')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

Problem Statement 4 Result

```
# Predicting outcomes using the trained RF Classifier
ypredicted = rand_for.predict(xtest)

# Training Score, Accuracy Score
print("Random Forest Traing Score: {}%".format(rand_for.score(xtrain, ytrain)*100))
print("Random Forest Accuracy Score: {}%".format(accuracy_score(ytest,
ypredicted)*100))
print('\n')
# Classification report of the RF classifier on the test set
print(classification_report(ytest, ypredicted))
```

FIGURE 12

```
# Adding column sevenormorepregnancies to the To_Predict dataset
topredict["SevenOrMorePregnancies"] = ''
# Looping through the dataset
for i in range(0, len(topredict)):
   if topredict["Pregnancies"][i] >= 7: # For number of pregnancies greater than or 
equal to 7
       topredict["SevenOrMorePregnancies"][i] = 1 # Setting the value to 1
   else:
       topredict["SevenOrMorePregnancies"][i] = 0

#Converting the datatype to integers
topredict["SevenOrMorePregnancies"] = topredict["SevenOrMorePregnancies"].astype(int)
# Addition of the PredictedOutcome column to the dataset
# Prediction of the outcome using the trained RF model
topredict["Predicted Outcome"] = rand_for.predict(topredict)
topredict # Display
```