DATA70121 Coursework: EDA & Regression

1. Brief description of the data

The “PimaDiabetes.csv” dataset contains the results of diabetes tests in 750 women along with eight diagnostic measures: Pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree, and Age. Below are the short descriptions of each diagnostic measure.

- Pregnancies: number of times the woman has been pregnant
- Glucose: plasma glucose concentration (mg/dl) at 2 hours in an oral glucose tolerance test (OGTT)
- Blood Pressure (“BloodPressure”): Diastolic blood pressure (mm Hg)
- Skin Thickness (“SkinThickness”): Triceps skin fold thickness (mm)
- Serum Insulin (“Insulin”): insulin concentration2 (μ U/ml) at 2 hours in an OGTT
- BMI: body mass index (weight in kg)/(height in m)2
- Diabete Pedigree (“DiabetePedigree”): a numerical score designed to measure the genetic influence of both the woman’s diabetic and her non‑diabetic relatives on diabetes risk: higher scores mean higher risk.
- Age: in years
- Outcome: 1 if the woman eventually tested positive for diabetes, zero otherwise

It is not medically possible to 0 values for the five measures; Glucose, Blood Pressure, Skin Thickness, Serum Insulin, and BMI. In addition, all eight measures have different measurements or ranges. Therefore, we need to discuss how to deal with these issues properly.

2. Exploratory data analysis

This section is to discover the statistical characteristics of the dataset, such as the structure, data types, distributions, and correlations between the variables. We can also check any missing values or duplicates. Then, we can discuss how to deal with some characteristics and which prediction model can be suitable.

The dataset consists of 750 rows and 9 columns. 'BMI' and 'DiabetesPedigree' are float numbers, and the others are integers. 'Non-Null Count' in Figure 1 and count numbers in Figure 2 show that there is no NULL value.

Figure 1. The brief information of the “PimaDiabetes” dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 750 entries, 0 to 749
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Pregnancies       750 non-null    int64
 1   Glucose           750 non-null    int64
 2   BloodPressure     750 non-null    int64
 3   SkinThickness     750 non-null    int64
 4   Insulin           750 non-null    int64
 5   BMI               750 non-null    float64
 6   DiabetesPedigree  750 non-null    float64
 7   Age               750 non-null    int64
 8   Outcome           750 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 52.9 KB
```

Figure 2. No. of NULL values in the “PimaDiabetes” dataset

```
diabetes_df.isnull().sum()
```

```
Pregnancies         0
Glucose             0
BloodPressure       0
SkinThickness       0
Insulin             0
BMI                 0
DiabetesPedigree    0
Age                 0
Outcome             0
dtype: int64
```

We also can confirm that there is no duplicate in the dataset as below.

Figure 3. Checking the existence of duplicates in the “PimaDiabetes” dataset

```
diabetes_df.duplicated().any()
```

```
False
```

The Figure 4 shows the first five rows of the dataset.

Figure 4. The first 5 rows in the "PimaDiabetes" dataset

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Figure 5 below shows statistics of the dataset. There are zero min values in 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', and 'BMI'. These values are medically impossible, so they have to be dealt with properly before model training. We can also find outliers in every column except 'Outcome' by looking at '25%(Q1)', '75%(Q3)', min, and max values.

Figure 5. Brief statistical analysis on the "PimaDiabetes" dataset

	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

In Figure 6, only 'Glucose' and 'BloodPressure' look normally distributed while the others are skewed. This implies that it is better to use medians to replace zero values in 'Insulin', 'BMI', and 'SkinThickness', while using mean values for 'Glucose' and 'BloodPressure'. This is because when a distribution is skewed, the mean would be misleading so the median is a better choice (Steinberg, 2010, p. 73). We also can find some outliers; boxplots will provide a clearer picture of this. It is also notable that 'Outcome' is imbalanced.

Figure 6. Histograms of the “PimaDiabetes” dataset

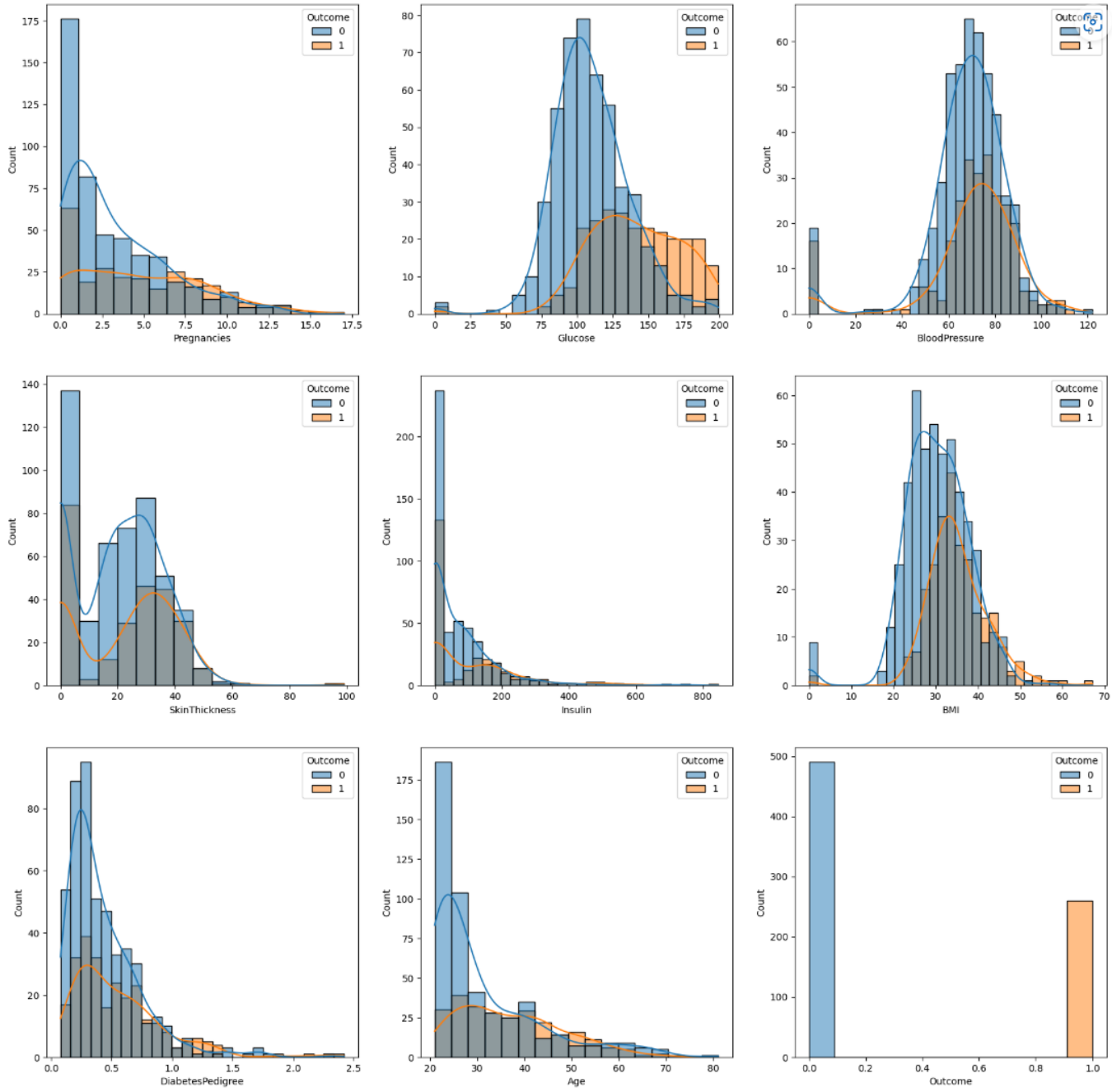


Figure 7 clearly shows the outliers. Care should be taken about how to deal with the outliers, especially in medical datasets like "PimaDiabetes.csv". Outliers can distort statistical analyses, but they may simply mean medical health problems (Deneshkumar, Senthamaraikannan and Manikandan, 2014, p. 243). Therefore, the outliers will not be replaced or deleted in this study.

Figure 7. Box plots of the "PimaDiabetes" dataset

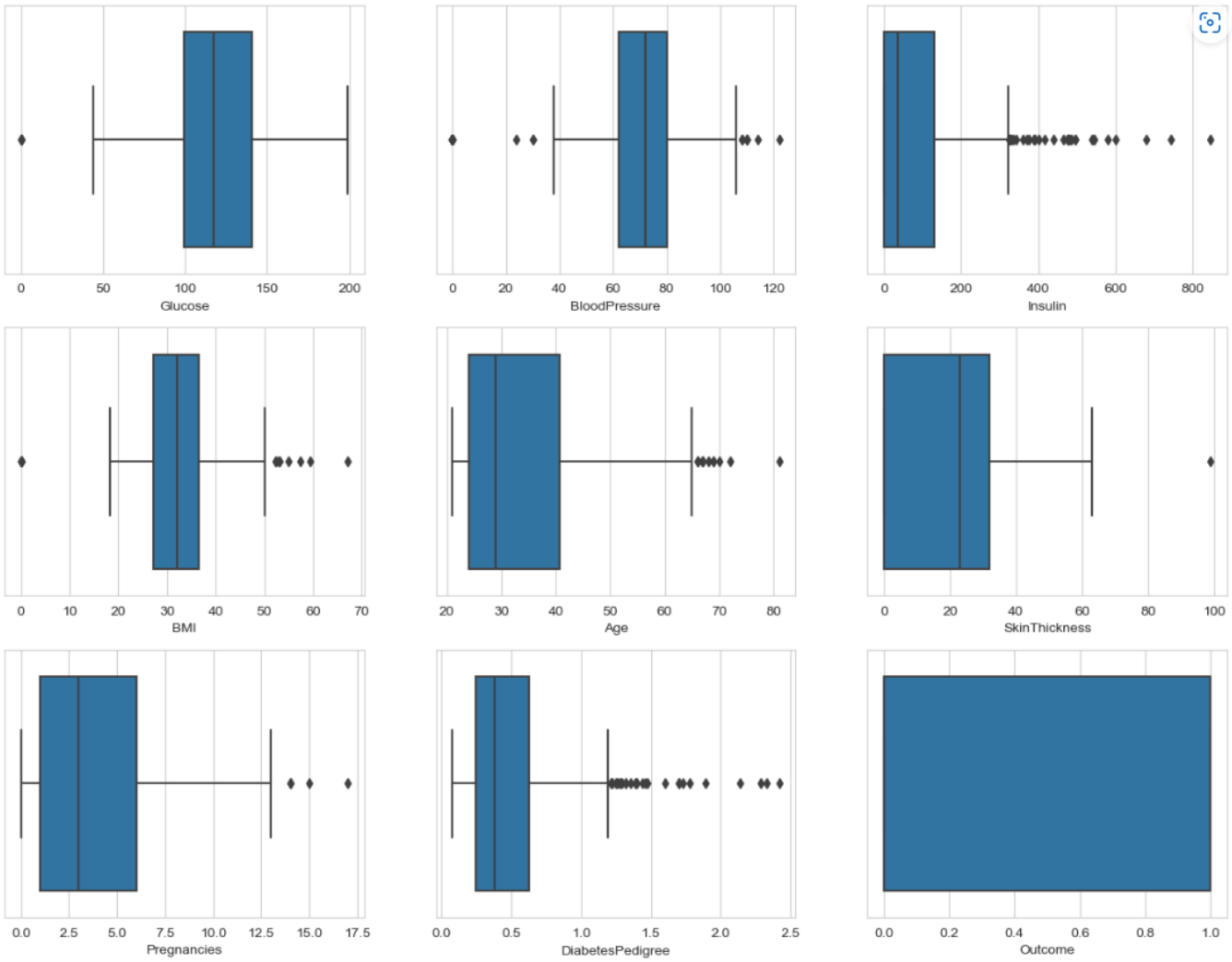


Figure 8. Pearson's correlation coefficients between the variables

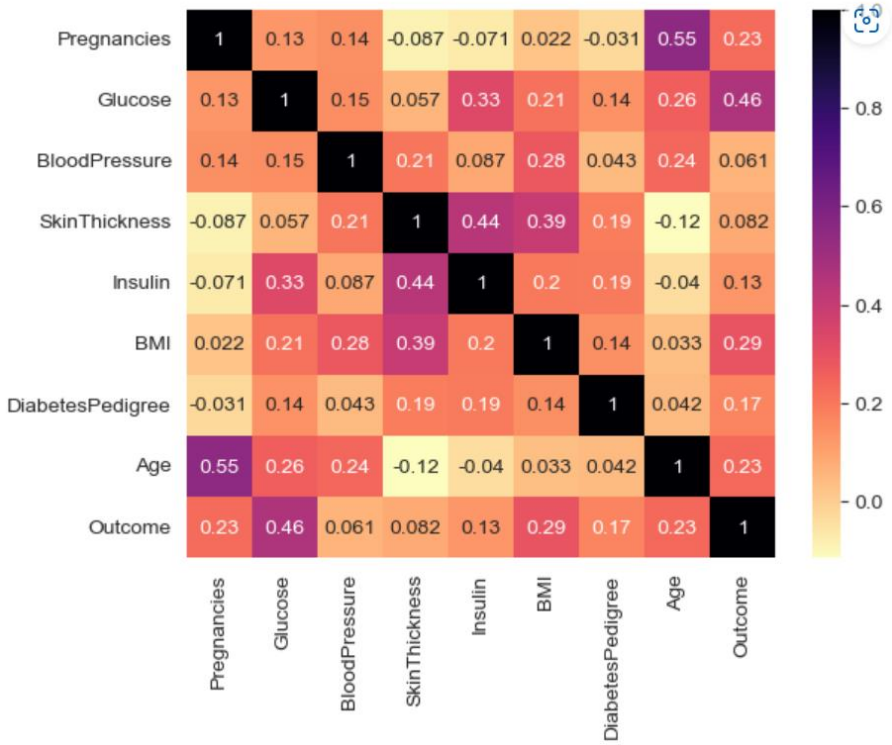


Figure 8 shows the Pearson correlation coefficients between the variables. We can see none of them has a high correlation with any others. To predict 'Outcome', only four variables seem to be useful considering the correlation coefficients: 'Pregnancies', 'Glucose', 'BMI', and 'Age'. The correlation coefficients of the other variables and 'Outcome' are less than 0.2, which is very low. We can do feature selection in Section 4 to see whether excluding the variables with low coefficients will bring higher prediction accuracy.

The zero values take up to 29.47% of the 'SkinThickness', and 48.26% of the 'Insulin'. In section 4, we also can see whether excluding these variables will bring better predictions by feature selection.

Figure 9. No. of zero values in 'BloodPressure', 'Glucose', 'SkinThickness', 'Insulin', and 'BMI'

```
no. of zeros in BloodPressure:35 (4.666666666666667%)
no. of zeros in Glucose:5 (0.6666666666666667%)
no. of zeros in SkinThickness:221 (29.46666666666667%)
no. of zeros in Insulin:362 (48.266666666666666%)
no. of zeros in BMI:11 (1.4666666666666666%)
```

Now we can replace zero values in 'Glucose', 'BloodPressure' by mean values, and the ones in 'SkinThickness', 'Insulin', and 'BMI' by medians. Figure 10 shows the result.

Figure 10. Brief analysis on the dataset after replacing zero values

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age	Outcome
count	750.000000	750.000000	750.000000	750.000000	750.000000	750.00000	750.000000	750.000000	750.000000
mean	3.844000	121.542249	72.201858	27.266667	97.996000	32.42840	0.473544	33.166667	0.346667
std	3.370085	30.451560	12.159438	9.241506	103.569424	6.90093	0.332119	11.708872	0.476226
min	0.000000	44.000000	24.000000	7.000000	14.000000	18.20000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	64.000000	23.000000	36.500000	27.50000	0.244000	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	36.750000	32.00000	0.377000	29.000000	0.000000
75%	6.000000	140.750000	80.000000	32.000000	129.750000	36.57500	0.628500	40.750000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.10000	2.420000	81.000000	1.000000

3. Regression by “ThreeOrMoreKids” predictor

We can see that the "ThreeOrMoreKids" were successfully added in Figure 11 and 12.

Figure 11. Brief analysis on the dataset after adding ‘ThreeOrMoreKids’

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age	ThreeOrMoreKids	Outcome
count	750.000000	750.000000	750.000000	750.000000	750.000000	750.00000	750.000000	750.000000	750.000000	750.000000
mean	3.844000	121.542249	72.201858	27.266667	97.996000	32.42840	0.473544	33.166667	0.546667	0.346667
std	3.370085	30.451560	12.159438	9.241506	103.569424	6.90093	0.332119	11.708872	0.498150	0.476226
min	0.000000	44.000000	24.000000	7.000000	14.000000	18.20000	0.078000	21.000000	0.000000	0.000000
25%	1.000000	99.000000	64.000000	23.000000	36.500000	27.50000	0.244000	24.000000	0.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	36.750000	32.00000	0.377000	29.000000	1.000000	0.000000
75%	6.000000	140.750000	80.000000	32.000000	129.750000	36.57500	0.628500	40.750000	1.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.10000	2.420000	81.000000	1.000000	1.000000

Figure 12. The first five rows of the dataset after adding ‘ThreeOrMoreKids’

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age	ThreeOrMoreKids	Outcome
0	6	148.0	72.0	35	36.5	33.6	0.627	50	1	1
1	1	85.0	66.0	29	36.5	26.6	0.351	31	0	0
2	8	183.0	64.0	23	36.5	23.3	0.672	32	1	1
3	1	89.0	66.0	23	94.0	28.1	0.167	21	0	0
4	0	137.0	40.0	35	168.0	43.1	2.288	33	0	1

Now we can train a logistic regression model using 'ThreeOrMoreKids' to predict 'Outcome'. Logistic regression was chosen because the 'Outcome' to predict is binary (0, 1).

Figure 13. Codes for training the logistic regression model

```
y=diabetes_df2['Outcome']
X=diabetes_df2[['ThreeOrMoreKids']]

logreg = LogisticRegression(class_weight='auto')

logreg.fit(X, y)
lr_pred=logreg.predict(X)
```

We can calculate the probability of a person getting diabetes, given that the person has two or fewer children, by using predict_proba function in sklearn. The result shows that the probability of getting diabetes is about 24.37%, given that the person has less than 3 kids. And the one given that the person has 3 or more kids is approximately 43.20%.

Figure 14. Codes to calculate probabilities of diabetes given 'ThreeOrMoreKids'

```
X0 = X[X['ThreeOrMoreKids'] == 0]
y_scores = logreg.predict_proba(X0)[:,1]
print(np.unique(y_scores))
```

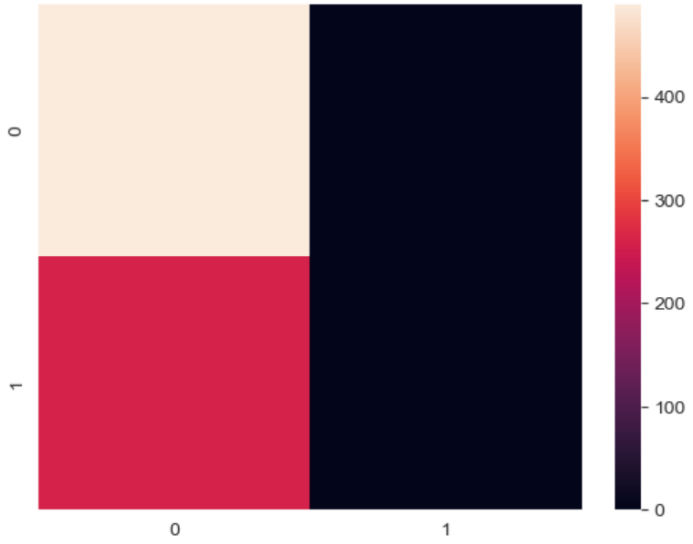
```
[0.24370277]
```

```
X1 = X[X['ThreeOrMoreKids'] == 1]
y_scores = logreg.predict_proba(X1)[:,1]
print(np.unique(y_scores))
```

```
[0.43205155]
```

The model always predicted 'Outcome' to be 1 because the probabilities of getting 1 are less than 0.5 (Figure 15).

Figure 15. Confusion matrix of the regression model



4. Multivariate Regression Model

Multivariate regression models will be trained to predict ‘Outcome’ for the new dataset ‘ToPredict.csv’ with standardisation and feature selection. We will find out the best combination of independent variables with the highest prediction accuracy. The best model will be used to predict ‘Outcome’ for the new dataset.

4.1. Standardisation

After standardising the dataset (Figure 16), we can check the dataset again in Figure 17, 18, and 19.

Figure 16. Codes to standardise the independent variables

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
diabetes_df_sc = diabetes_df.copy(deep=True)
diabetes_df_sc.iloc[:,0:8] = scaler.fit_transform(diabetes_df_sc.iloc[:,0:8])
```

Figure 17. Brief analysis on the standardised dataset

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age	Outcome
count	7.500000e+02	7.500000e+02	7.500000e+02	7.500000e+02	7.500000e+02	7.500000e+02	7.500000e+02	7.500000e+02	750.000000
mean	-1.406282e-17	-5.595524e-16	-9.978685e-16	4.019007e-17	1.310063e-16	5.598485e-16	-9.015011e-17	1.912544e-16	0.346667
std	1.000667e+00	1.000667e+00	1.000667e+00	1.000667e+00	1.000667e+00	1.000667e+00	1.000667e+00	1.000667e+00	0.476226
min	-1.141385e+00	-2.548112e+00	-3.966797e+00	-2.194468e+00	-8.115528e-01	-2.063185e+00	-1.191764e+00	-1.039792e+00	0.000000
25%	-8.444587e-01	-7.407598e-01	-6.749762e-01	-4.619933e-01	-5.941622e-01	-7.146412e-01	-6.916101e-01	-7.834046e-01	0.000000
50%	-2.506059e-01	-1.492626e-01	-1.661199e-02	-4.619933e-01	-5.917468e-01	-6.212002e-02	-2.908846e-01	-3.560930e-01	0.000000
75%	6.401734e-01	6.311850e-01	6.417522e-01	5.125238e-01	3.068009e-01	6.012765e-01	4.668784e-01	6.480892e-01	1.000000
max	3.906364e+00	2.545336e+00	4.098164e+00	7.767263e+00	7.227067e+00	5.027545e+00	5.864621e+00	4.087947e+00	1.000000

Figure 18. Pearson’s correlation coefficients between the variables after standardisation

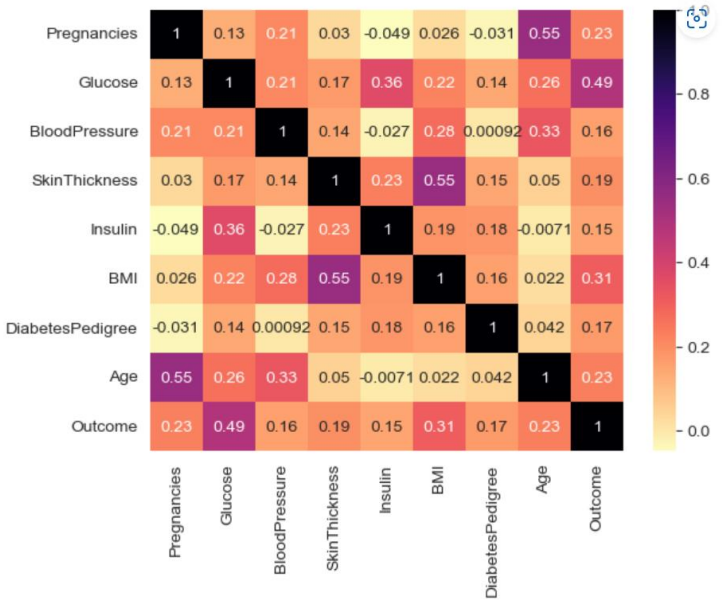
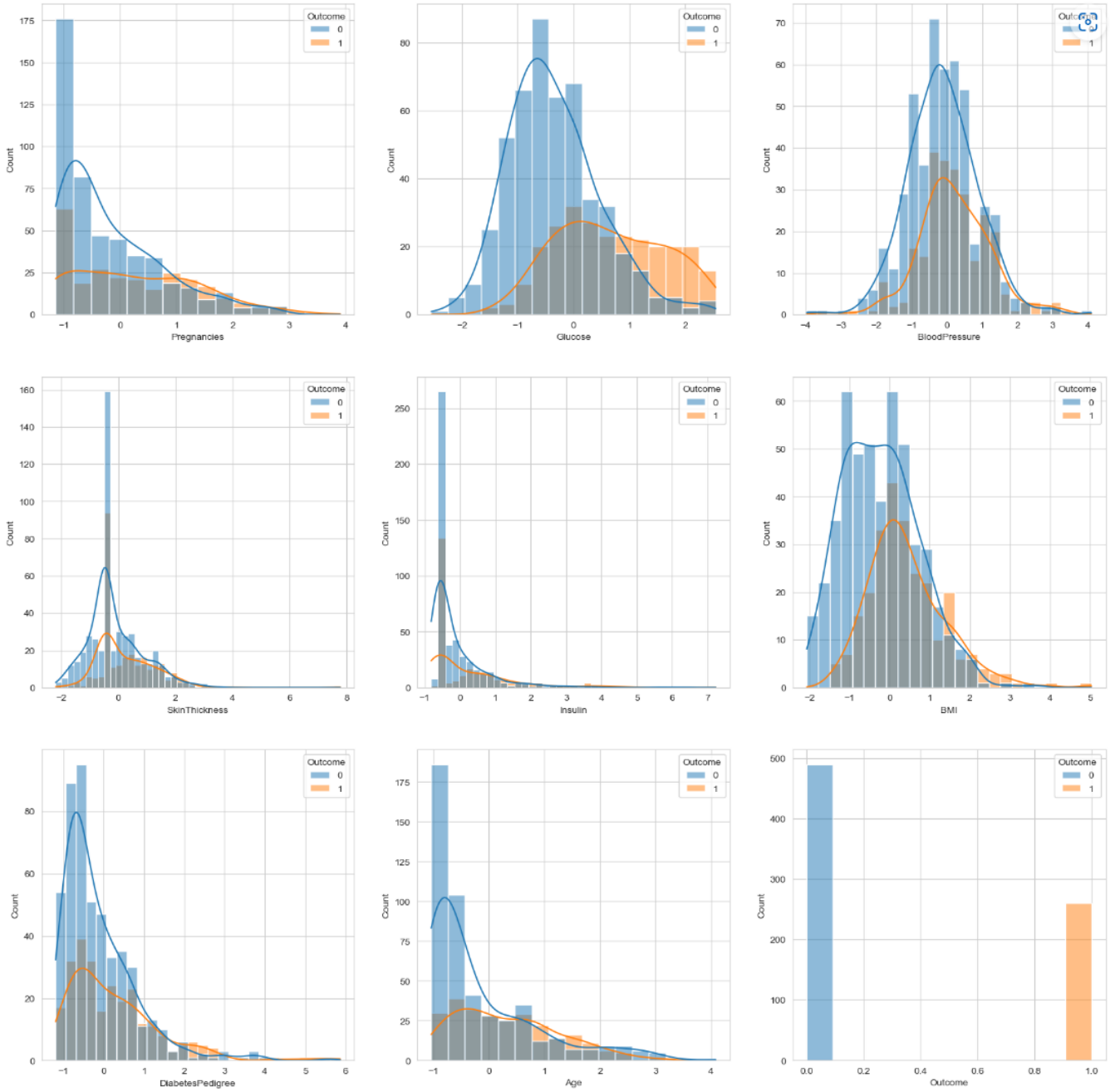


Figure 19. Histograms after standardisation



4.2. Feature selection

As mentioned in section 2, we can try various combinations of independent variables to train our models.

- X_fs0: obtained by replacing medically impossible zero values in 'Glucose', 'BloodPressure', 'SkinThickness', and 'BMI' in the original dataset
- X_fs1: obtained by standardising X_fs0
- X_fs2 ~ X_fs9: obtained by dropping a variable with the smallest Pearson correlation coefficients (Figure 18) with 'Outcome' from the previous combination

Figure 20. Codes for feature selection

```
X_fs0 = diabetes_df.iloc[:, 0:8] #dataset only after replacing 0 values that are medically impossible
X_fs1 = diabetes_df_sc.iloc[:, 0:8] #dataset after standardising X_fs0
X_fs2 = diabetes_df_sc[['Pregnancies','Glucose','BloodPressure', 'SkinThickness', 'BMI','DiabetesPedigree', 'Age']]
X_fs3 = diabetes_df_sc[['Pregnancies','Glucose','SkinThickness', 'BMI','DiabetesPedigree', 'Age']]
X_fs4 = diabetes_df_sc[['Pregnancies','Glucose','SkinThickness', 'BMI', 'Age']]
X_fs5 = diabetes_df_sc[['Pregnancies','Glucose','BMI','Age']]
X_fs6 = diabetes_df_sc[['Pregnancies', 'Glucose', 'BMI']]
X_fs7 = diabetes_df_sc[['Pregnancies', 'Glucose']]
X_fs8 = diabetes_df_sc[['Pregnancies']]
```

4.3. Model training and choosing the best model

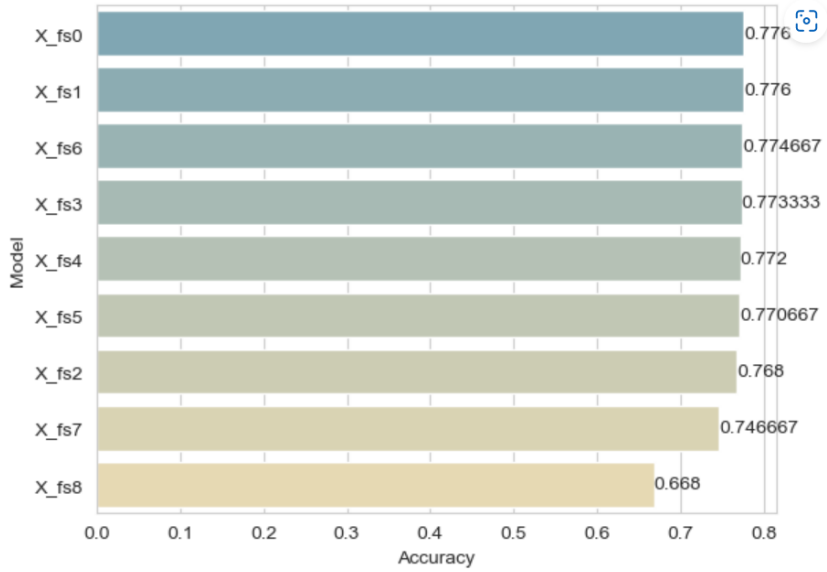
LogisticRegression model was trained by X_fs0 ~ X_fs9 as below.

Figure 21. Codes for model training

```
logreg = LogisticRegression(max_iter = 10000)

y = diabetes_df_sc['Outcome']
# fit the model with data
logreg.fit(X_fs0, diabetes_df['Outcome'])
lr_pred0=logreg.predict(X_fs0)
acc0 =logreg.score(X_fs0, y)

logreg.fit(X_fs1, y)
lr_pred1=logreg.predict(X_fs1)
acc1 =logreg.score(X_fs1, y)

logreg.fit(X_fs2, y)
lr_pred2=logreg.predict(X_fs2)
acc2 =logreg.score(X_fs2, y)

logreg.fit(X_fs3, y)
lr_pred3=logreg.predict(X_fs3)
acc3 =logreg.score(X_fs3, y)

logreg.fit(X_fs4, y)
lr_pred4=logreg.predict(X_fs4)
acc4 =logreg.score(X_fs4, y)

logreg.fit(X_fs5, y)
lr_pred5=logreg.predict(X_fs5)
acc5 =logreg.score(X_fs5, y)

logreg.fit(X_fs6, y)
lr_pred6=logreg.predict(X_fs6)
acc6 =logreg.score(X_fs6, y)

logreg.fit(X_fs7, y)
lr_pred7=logreg.predict(X_fs7)
acc7 =logreg.score(X_fs7, y)

logreg.fit(X_fs8, y)
lr_pred8=logreg.predict(X_fs8)
acc8 =logreg.score(X_fs8, y)
```

X_fs0 and X_fs1 showed the highest accuracy for prediction, followed by X_fs6 (Figure 22). This means that the model using all the independent variables is the best, and the one using the first four variables with the highest Pearson correlation coefficient is the second best.

Figure 22. Comparing the accuracies of the models



Meanwhile, having no difference between X_fs0 and X_fs1 means that standardising makes no difference in this study.

4.4. Predict and interpret the result using ToPredict.csv

There are zero values in 'SkinThickness' and 'Insulin' in the new dataset ('ToPredict.csv'). Therefore, we need to replace them with the medians from the original dataset ('PimaDiabetes.csv').

Figure 23. The 'ToPredict.csv' dataset

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age
0	4	136	70	0	0	31.2	1.182	22
1	1	121	78	39	74	39.0	0.261	28
2	3	108	62	24	0	26.0	0.223	25
3	0	181	88	44	510	43.3	0.222	26
4	8	154	78	32	0	32.4	0.443	45

Figure 24. The 'ToPredict.csv' dataset after replacing zero values

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age
0	4	136	70	23	36.75	31.2	1.182	22
1	1	121	78	39	74.00	39.0	0.261	28
2	3	108	62	24	36.75	26.0	0.223	25
3	0	181	88	44	510.00	43.3	0.222	26
4	8	154	78	32	36.75	32.4	0.443	45

Now we can predict the 'Outcome' by the prediction model trained by X_fs0 (no standardisation, after replacing zero values in the original dataset).

Figure 25. Predicting the ‘Outcome’ using ‘ToPredict.csv’

```
logreg = LogisticRegression(max_iter = 10000)
logreg.fit(X_fs0, diabetes_df['Outcome'])
X_final = to_predict_df.iloc[:, 0:8]
lr_pred_final = logreg.predict(X_final)
lr_pred_final
```

```
array([1, 0, 0, 1, 1], dtype=int64)
```

The predicted 'Outcome' are 1, 0, 0, 1, 1 (Positive, Negative, Negative, Positive, Positive).

Figure 26. Probabilities of getting ‘Outcome 1’

```
y_scores_final = logreg.predict_proba(X_final)[:,1]
print(y_scores_final)
```

```
[0.53420836 0.297994   0.09925573 0.71153538 0.74133756]
```

The probabilities of getting 'Outcome' 1 (diabetes) are 53.42%, 29.80%, 9.93%, 71.15%, 74.13%, respectively.

5. Codes in Python

```
#Import packages
import numpy as np
import pandas as pd

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.metrics import f1_score, precision_score, recall_score,accuracy_score
from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt
import seaborn as sns

diabetes_df = pd.read_csv('PimaDiabetes.csv')

#2. Exploratory Data Analysis
diabetes_df.shape
diabetes_df.info()
diabetes_df.isnull().sum()
diabetes_df.duplicated().any()
diabetes_df.head()
diabetes_df.describe()

#Histograms
colNames = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
'BMI', 'DiabetesPedigree', 'Age', 'Outcome']
x_axis = ['no. of times', 'mg/dl', 'mm Hg', 'mm', 'U/ml', '(weight in kg)/(height in
m)^2', 'score', 'age in years', 'Outcome']

fig, axs = plt.subplots(3,3, figsize=(20,20))
row = 0
col = 0

for i, x in enumerate(colNames):
    sns.histplot(data=diabetes_df, x=colNames[i], kde=True, ax = axs[row, col],
hue=diabetes_df['Outcome'])
    col += 1
    if col > 2:
        row += 1
        col = 0

#box plots
plt.figure(figsize=(16,12))
sns.set_style(style='whitegrid')
plt.subplot(3,3,1)
sns.boxplot(x='Glucose',data=diabetes_df)
plt.subplot(3,3,2)
sns.boxplot(x='BloodPressure',data=diabetes_df)
plt.subplot(3,3,3)
sns.boxplot(x='Insulin',data=diabetes_df)
plt.subplot(3,3,4)
sns.boxplot(x='BMI',data=diabetes_df)
```

```
plt.subplot(3,3,5)
sns.boxplot(x='Age',data=diabetes_df)
plt.subplot(3,3,6)
sns.boxplot(x='SkinThickness',data=diabetes_df)
plt.subplot(3,3,7)
sns.boxplot(x='Pregnancies',data=diabetes_df)
plt.subplot(3,3,8)
sns.boxplot(x='DiabetesPedigree',data=diabetes_df)
plt.subplot(3,3,9)
sns.boxplot(x='Outcome',data=diabetes_df)

#Correlation heatmap
sns.heatmap(diabetes_df.corr(), annot = True, cmap = 'magma_r');

#count zero values
colNames2 = ['BloodPressure', 'Glucose', 'SkinThickness', 'Insulin', 'BMI']
for i, x in enumerate(colNames2):
    z = diabetes_df[diabetes_df[x] == 0].shape[0]
    total = diabetes_df[x].count()
    print('no. of zeros in ', colNames2[i], ':', z, ' (', z/total*100, '%)', sep='')

#replace zero values
diabetes_df['Glucose']=diabetes_df['Glucose'].replace(0,diabetes_df['Glucose'].mean())
diabetes_df['BloodPressure']=diabetes_df['BloodPressure'].replace(0,diabetes_df['Blood
Pressure'].mean())

diabetes_df['SkinThickness']=diabetes_df['SkinThickness'].replace(0,diabetes_df['SkinT
hickness'].median())
diabetes_df['Insulin']=diabetes_df['Insulin'].replace(0,diabetes_df['Insulin'].median(
))
diabetes_df['BMI']=diabetes_df['BMI'].replace(0,diabetes_df['BMI'].median())
diabetes_df.describe()

#3. Univariate Regression by “ThreeOrMoreKids”
diabetes_df2 = diabetes_df.iloc[:,0:8].copy(deep=True)
diabetes_df2['ThreeOrMoreKids'] = np.where(diabetes_df['Pregnancies'] > 2, 1, 0)
diabetes_df2['Outcome'] = diabetes_df['Outcome'].copy(deep=True)
diabetes_df2.describe()

diabetes_df2.head()

#Train regression model
y=diabetes_df2['Outcome']
X=diabetes_df2[['ThreeOrMoreKids']]

logreg = LogisticRegression(class_weight='auto')

logreg.fit(X, y)
lr_pred=logreg.predict(X)

#Calculate probabilities
X0 = X[X['ThreeOrMoreKids'] == 0]
y_scores = logreg.predict_proba(X0)[:,1]
```

```
print(np.unique(y_scores))
X1 = X[X['ThreeOrMoreKids'] == 1]
y_scores = logreg.predict_proba(X1)[:,1]
print(np.unique(y_scores))

#Classification report with confusion matrix
print("Classification Report is:\n",classification_report(y,lr_pred))
print("\n Confusion Matrix:\n")
sns.heatmap(confusion_matrix(y,lr_pred))

#4. Multivariate Regression Model
#4.1. Standardisation
scaler = StandardScaler()
diabetes_df_sc = diabetes_df.copy(deep=True)
diabetes_df_sc.iloc[:,0:8] = scaler.fit_transform(diabetes_df_sc.iloc[:,0:8])
diabetes_df_sc.describe()

#correlation heatmap
sns.heatmap(diabetes_df_sc.corr(), annot = True, cmap = 'magma_r');

#histograms
colNames = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
'BMI', 'DiabetesPedigree', 'Age', 'Outcome']
x_axis = ['no. of times', 'mg/dl', 'mm Hg', 'mm', 'U/ml', '(weight in kg)/(height in
m)^2', 'score', 'age in years', 'Outcome']

fig, axs = plt.subplots(3,3, figsize=(20,20))
row = 0
col = 0

for i, x in enumerate(colNames):
    sns.histplot(data=diabetes_df_sc, x=colNames[i], kde=True, ax = axs[row, col],
hue=diabetes_df_sc['Outcome'])
    col += 1
    if col > 2:
        row += 1
        col = 0

#4.2. Feature selection
X_fs0 = diabetes_df.iloc[:, 0:8] #dataset only after replacing 0 values that are
medically impossible
X_fs1 = diabetes_df_sc.iloc[:, 0:8] #dataset after standardising X_fs0
X_fs2 = diabetes_df_sc[['Pregnancies','Glucose','BloodPressure', 'SkinThickness',
'BMI','DiabetesPedigree', 'Age']]
X_fs3 = diabetes_df_sc[['Pregnancies','Glucose','SkinThickness',
'BMI','DiabetesPedigree', 'Age']]
X_fs4 = diabetes_df_sc[['Pregnancies','Glucose','SkinThickness', 'BMI', 'Age']]
X_fs5 = diabetes_df_sc[['Pregnancies','Glucose','BMI','Age']]
X_fs6 = diabetes_df_sc[['Pregnancies', 'Glucose', 'BMI']]
X_fs7 = diabetes_df_sc[['Pregnancies', 'Glucose']]
X_fs8 = diabetes_df_sc[['Pregnancies']]

#4.3. Model training
```

```
logreg = LogisticRegression(max_iter = 10000)

y = diabetes_df_sc['Outcome']
# fit the model with data
logreg.fit(X_fs0, diabetes_df['Outcome'])
lr_pred0=logreg.predict(X_fs0)
acc0 =logreg.score(X_fs0, y)

logreg.fit(X_fs1, y)
lr_pred1=logreg.predict(X_fs1)
acc1 =logreg.score(X_fs1, y)

logreg.fit(X_fs2, y)
lr_pred2=logreg.predict(X_fs2)
acc2 =logreg.score(X_fs2, y)

logreg.fit(X_fs3, y)
lr_pred3=logreg.predict(X_fs3)
acc3 =logreg.score(X_fs3, y)

logreg.fit(X_fs4, y)
lr_pred4=logreg.predict(X_fs4)
acc4 =logreg.score(X_fs4, y)

logreg.fit(X_fs5, y)
lr_pred5=logreg.predict(X_fs5)
acc5 =logreg.score(X_fs5, y)

logreg.fit(X_fs6, y)
lr_pred6=logreg.predict(X_fs6)
acc6 =logreg.score(X_fs6, y)

logreg.fit(X_fs7, y)
lr_pred7=logreg.predict(X_fs7)
acc7 =logreg.score(X_fs7, y)

logreg.fit(X_fs8, y)
lr_pred8=logreg.predict(X_fs8)
acc8 =logreg.score(X_fs8, y)

#Compare the results
models = pd.DataFrame({'Model': ['X_fs0','X_fs1', 'X_fs2', 'X_fs3', 'X_fs4', 'X_fs5',
'X_fs6', 'X_fs7', 'X_fs8'],
    'Accuracy': [acc0, acc1, acc2, acc3, acc4, acc5, acc6, acc7, acc8]})
models_sorted = models.sort_values(by='Accuracy', ascending=False)
ax = sns.barplot(x = models_sorted['Accuracy'], y = models_sorted['Model'],
palette="blend:#7AB,#EDA");
ax.bar_label(ax.containers[0])

#4.4. Predict and interpret the result using ToPredict.csv
to_predict_df = pd.read_csv('ToPredict.csv')
to_predict_df
```

```
#Replace zero values
to_predict_df['Glucose']=to_predict_df['Glucose'].replace(0,diabetes_df['Glucose'].mea
n())
to_predict_df['BloodPressure']=to_predict_df['BloodPressure'].replace(0,diabetes_df['B
loodPressure'].mean())

to_predict_df['SkinThickness']=to_predict_df['SkinThickness'].replace(0,diabetes_df['S
kinThickness'].median())
to_predict_df['Insulin']=to_predict_df['Insulin'].replace(0,diabetes_df['Insulin'].med
ian())
to_predict_df['BMI']=to_predict_df['BMI'].replace(0,diabetes_df['BMI'].median())

to_predict_df

#Predict outcomes using model0
logreg = LogisticRegression(max_iter = 10000)
logreg.fit(X_fs0, diabetes_df['Outcome'])
X_final = to_predict_df.iloc[:, 0:8]
lr_pred_final = logreg.predict(X_final)
lr_pred_final
y_scores_final = logreg.predict_proba(X_final)[:,1]
print(y_scores_final)
```

REFERENCES

- Deneshkumar, V., Senthamaraikannan, K. and Manikandan, M. (2014). ‘Identification of Outliers in Medical Diagnostic System Using Data Mining Techniques’. International Journal of Statistics and Applications 2014, 4(6), pp. 241-248. Available at: DOI: 10.5923/j.statistics.20140406.01 (Accessed: 16 November 2022).

- NHS (2022). What is blood pressure?. Available at: https://www.nhs.uk/common-health-questions/lifestyle/what-is-blood-pressure/ (Accessed: 16 November 2022).

- Steinberg, W. (2010). *Statistics Alive!* 2nd ed. California: SAGE.