Pima Indian Diabetes
EDA & Regression

1. Introduction of PimaDiabetes Dataset

The dataset is from The National Institute of Diabetes and Digestive and Kidney Diseases. Its main goal is to help in determining if a patient is diabetic, using various measures. This data was collected from the Pima Indian community in Phoenix, Arizona, USA who has been the focus of research due to their high susceptibility to diabetes.

Each entry in the dataset represents a medical examination of a female Pima Indian. The dataset comprises eight different input variables and one response. Smith et al., (1988) described the variables as follows:

- `Pregnancies` : Number of times pregnant
- `Glucose` : Plasma glucose concentration (mg/dl) at 2 hours in an Oral Glucose Tolerance Test (GTT)
- `Blood Pressure` : Diastolic blood pressure (mm Hg)
- `Skin Thickness` : Triceps skin fold thickness (mm)
- `Insulin` : Insulin concentration (μ U/ml) at 2 hours in an OGTT
- `BMI` : Body mass index (weight in kg)/(height in m)^2
- `Diabetes Pedigree` : The genetic influence of the subject's relatives with condition of both affected and unaffected to diabetes.
- `Age` : Age (years)
- `Outcome` : Has diabetes (1) or not (0)

Additionally, there is issue with data quality as the dataset contains multiple zero values in measurements such as `SkinThickness` and `Insulin` levels, which are implausible.

2. Exploratory Data Analysis (EDA)

EDA aims to provide a statistical overview, explore the distribution of data, and understand the relationships between variables. The summary statistics for each feature are as follows:

Table 1: Summary Statistics of Features

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

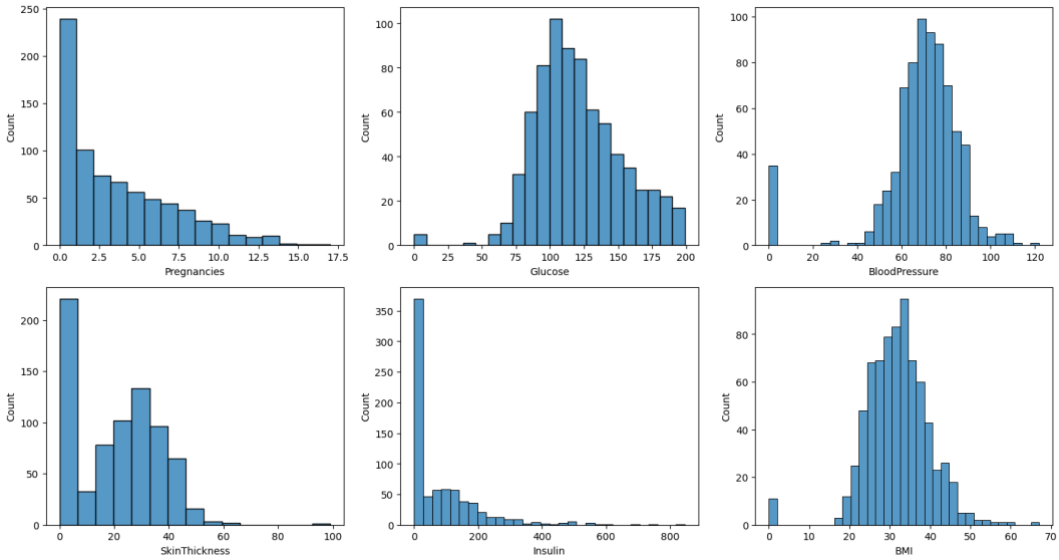
The summary reveals that several features have zero values. These zeroes are unlikely to be accurate and may represent missing data inputted as zero. The count of zero-value records for each of these features is presented in Table 2.

Table 2: Zero Values Count

Pregnancies	109
Glucose	5
BloodPressure	35
SkinThickness	221
Insulin	362
BMI	11
DiabetesPedigree	0
Age	0
dtype: int64	

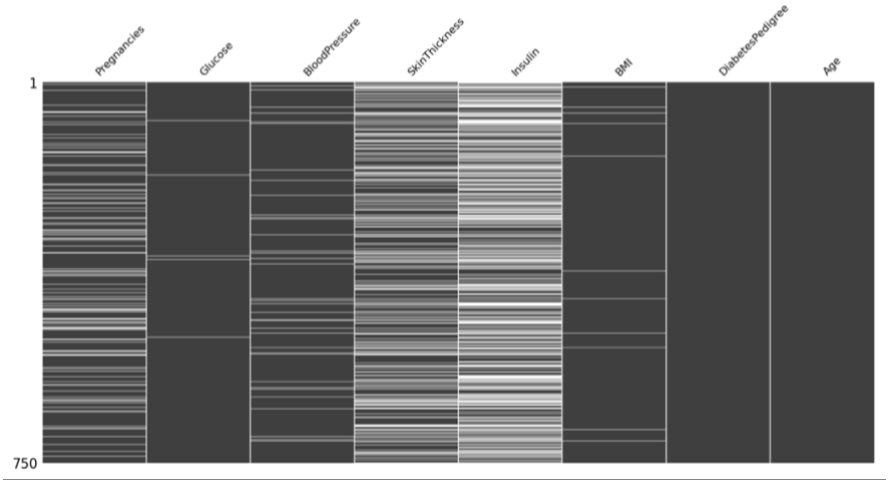
Most of the features have unlikely zero values, especially `SkinThickness` and `Insulin`, which have nearly 30% and 50% of zero values of the records. Meanwhile, zero values in the `Pregnancies` feature are plausible and can occur in the observation. However, the presence of zero in the other variables raises uncertainty about the validity of having zero `Pregnancies`. The impact of these zero values on the data distribution is illustrated in Figure 1.

Figure 1: Histogram of Features with Zero Values



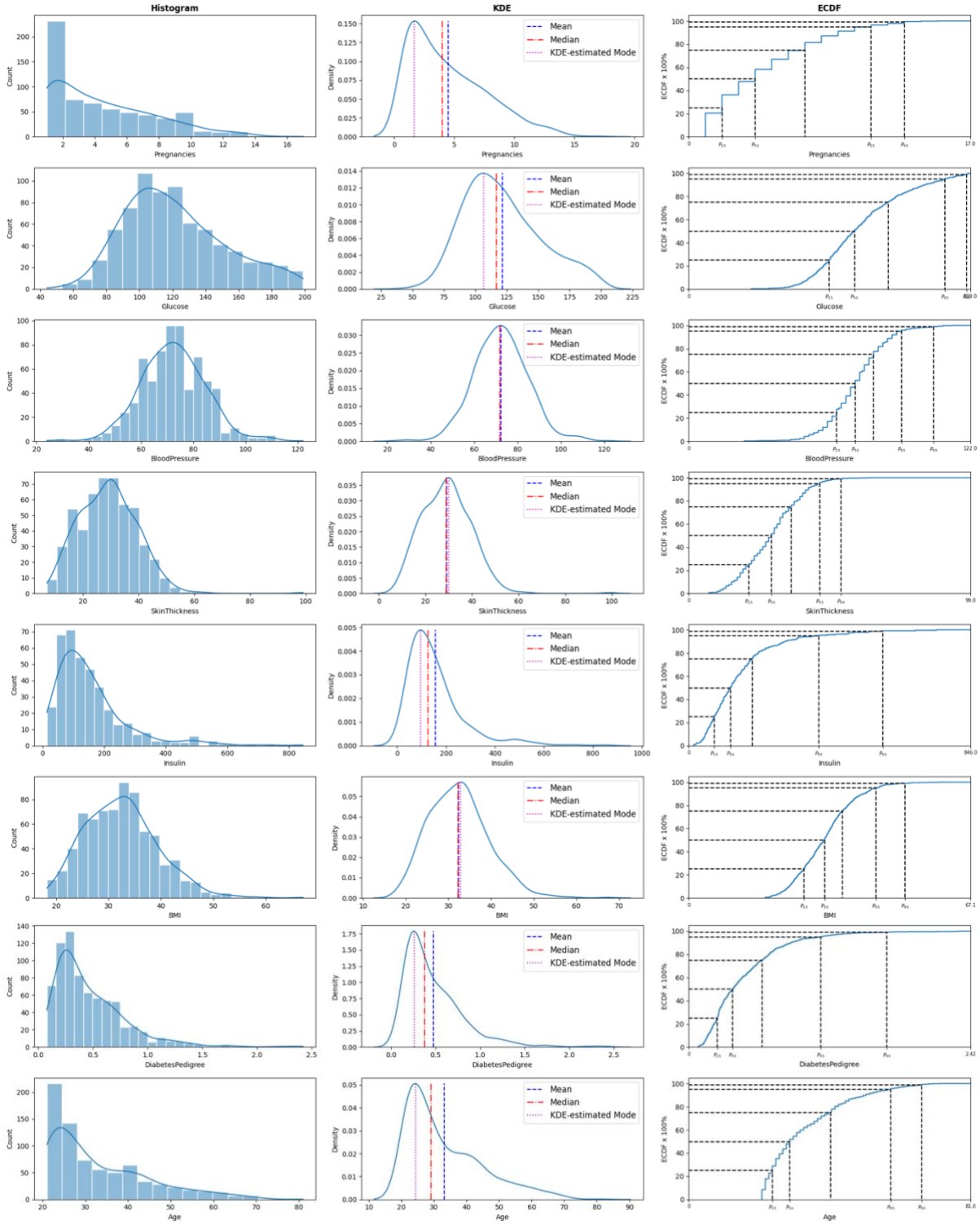
The zero values seem to shift the distribution. Given the improbability of zero in these observations, they were treated as missing. Next, the *missingno* library (Bilogur, 2023) was utilized to visualize and examine the pattern of these missing values.

Figure 2: Missing Values Matrix



The resulting matrix indicates that the missing (or zero) values are distributed randomly across features, as there is no discernible pattern linking them. Subsequently, each feature was visualized in histogram, KDE, and ECDF to show the distribution of the data without the missing values.

Figure 3: Histogram, KDE, ECDF of Each Feature



This examination reveals that most features are right-skewed, except for `BloodPressure`, `SkinThickness`, and `BMI`, which show a normal distribution. As a next step, the interaction and correlation between features and response variables were depicted in Figure 4 and 5.

Figure 4: Multivariate EDA

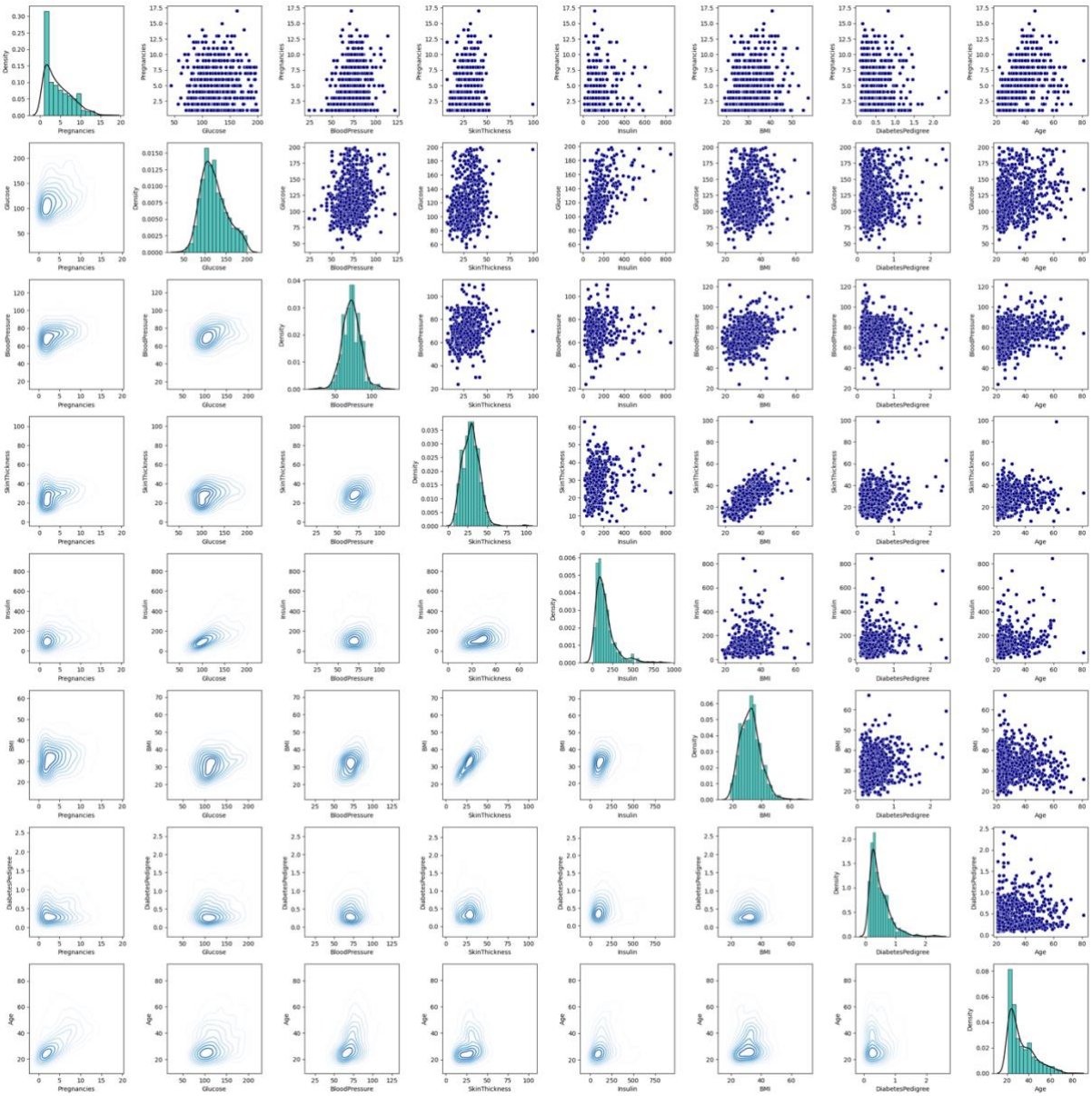
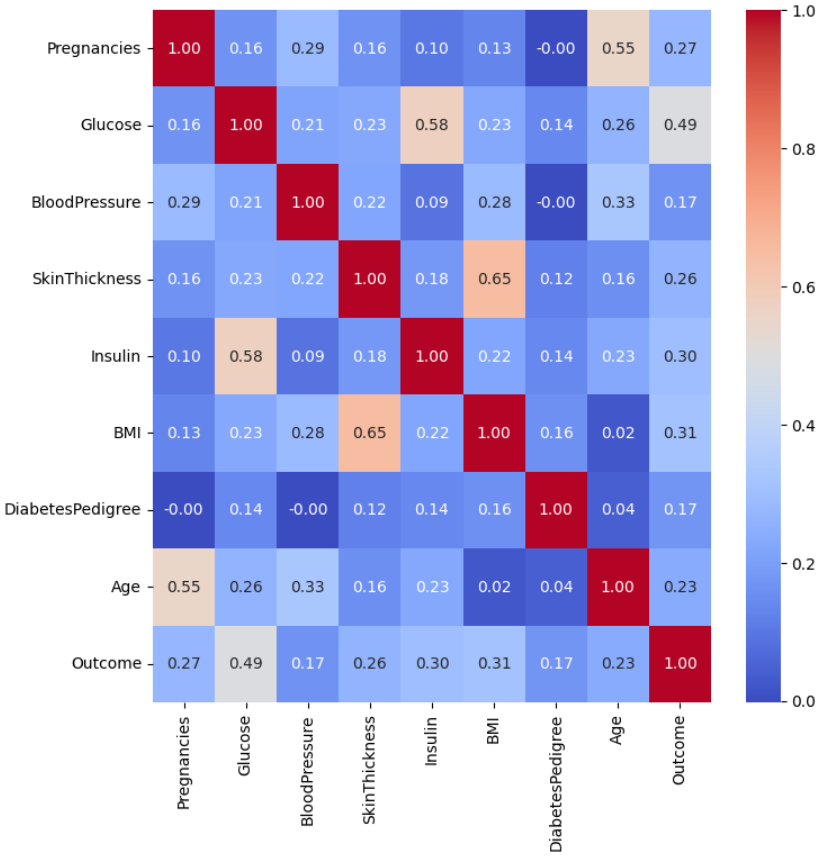


Figure 5: Correlation Heatmap



Figures 4 and 5 suggest moderate to strong correlation between `BMI` and `SkinThickness`, and between `Glucose` and `Insulin`. Since `SkinThickness` and `Insulin` have plenty of missing entries, they can be replaced by `BMI` and `Glucose` in the regression model. In addition, `BloodPressure` and `DiabetesPedigree` appear to have weak correlation with the outcome, suggesting they may not significantly contribute to diabetes prediction.

3. `SevenOrMorePregnancies` as Single Predictor

`SevenOrMorePregnancies` was constructed based on the number of `Pregnancies`, which have value of one if number of `Pregnancies` is more or equal to seven, and zero otherwise. Given the binomial response, a logistic regression model was developed using `SevenOrMorePregnancies` as the sole predictor, employing the *Logit* function from Python's *statsmodels*.

Table 3: Logistic Regression Results of *SevenOrMorePregnancies*

Results: Logit

Model:	Logit	Method:	MLE
Dependent Variable:	Outcome	Pseudo R-squared:	0.045
Date:	2023-11-14 11:25	AIC:	928.7772
No. Observations:	750	BIC:	938.0173
Df Model:	1	Log-Likelihood:	-462.39
Df Residuals:	748	LL-Null:	-484.02
Converged:	1.0000	LLR p-value:	4.7906e-11
No. Iterations:	5.0000	Scale:	1.0000

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
Intercept	-0.9199	0.0915	-10.0519	0.0000	-1.0992	-0.7405
SevenOrMorePregnancies	1.1898	0.1822	6.5287	0.0000	0.8326	1.5470

▪ The probability of diabetes occurrence in subjects with six or fewer pregnancies:

```
odds6 = math.exp(logitModel.params[0] + logitModel.params[1] * 0)
prob6 = odds6 / (1 + odds6)
print(f'Probability of developing diabetes, given six or fewer pregnancies is {prob6:.2f}')
```

```
Probability of developing diabetes, given six or fewer pregnancies is 0.28
```

▪ The probability when the subject has had seven or more pregnancies:

```
odds7 = math.exp(logitModel.params[0] + logitModel.params[1] * 1)
prob7 = odds7 / (1 + odds7)
print(f'Probability of developing diabetes, given seven or more pregnancies is {prob7:.2f}')
```

```
Probability of developing diabetes, given seven or more pregnancies is 0.57
```

4. Modelling with The Other Features

Prior to model development with additional features, it was important to address missing values. Given the substantial number of missing entries in `Insulin` and `SkinThickness`, and considering the adequate representation by `Glucose` and `BMI`, these features were excluded from the model. Moreover, `SevenOrMorePregnancies` was preferred over the `Pregnancies` due to issues with zero values. For imputing missing values in `Glucose`, `BloodPressure`, and `BMI`, the k-Nearest Neighbour method was applied (Troyanskaya et al., 2001).

Various models were then created using a combination of `Glucose`, `BMI`, `Age`, and `SevenOrMorePregnancies`, excluding `BloodPressure` and `DiabetesPedigree` due to their low correlation with the outcome. Each model was evaluated and compared based on their AIC values.

Table 4: Summary of Model Combination

No.	Model	AIC
1	Outcome ~ Glucose	738.59
2	Outcome ~ BMI	898.47
3	Outcome ~ Age	932.22
4	Outcome ~ SevenOrMorePregnancies	928.77
5	Outcome ~ Glucose + BMI	744.68
6	Outcome ~ Glucose + Age	775.66
7	Outcome ~ Glucose + SevenOrMorePregnancies	758.36
8	Outcome ~ BMI + Age	858.09
9	Outcome ~ BMI + SevenOrMorePregnancies	858.96
10	Outcome ~ Age + SevenOrMorePregnancies	917.08
11	Outcome ~ Glucose + BMI + Age	733.25
12	Outcome ~ Glucose + BMI + SevenOrMorePregnancies	718.77
13	Outcome ~ Glucose + Age + SevenOrMorePregnancies	759.47
14	Outcome ~ BMI + Age + SevenOrMorePregnancies	844.97
15	Outcome ~ Glucose + BMI + Age + SevenOrMorePregnancies	718.53

The best-performing model, based on the lowest AIC, includes all four features. However, the model number 12 with just three features shows insignificant difference in AIC values. To maintain simplicity and prevent overfitting, the three-feature model was selected with following results:

Table 5: Logistic Regression Results of Selected Model

Results: Logit

Model:	Logit	Method:	MLE
Dependent Variable:	Outcome	Pseudo R-squared:	0.266
Date:	2023-11-13 15:45	AIC:	718.7730
No. Observations:	750	BIC:	737.2533
Df Model:	3	Log-Likelihood:	-355.39
Df Residuals:	746	LL-Null:	-484.02
Converged:	1.0000	LLR p-value:	1.7555e-55
No. Iterations:	6.0000	Scale:	1.0000

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
Intercept	-8.4166	0.6712	-12.5399	0.0000	-9.7321	-7.1011
Glucose	0.0365	0.0035	10.5106	0.0000	0.0297	0.0433
BMI	0.0889	0.0145	6.1360	0.0000	0.0605	0.1173
SevenOrMorePregnancies	1.1243	0.2143	5.2455	0.0000	0.7042	1.5444

Following this, the chosen model was then used to predict diabetes development of 5 subjects in *ToPredict.* The predictions are presented in the `DiabetesPrediction` column in Table 6.

Table 6: Prediction Results

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

In the preferred model, `Glucose`, `BMI`, and `SevenOrMorePregnancies` have distinct contribution to the probability of developing diabetes. The observation in the first row was used to understand individual contribution by changing one variable and keeping the others constant. An illustration of `Glucose` variable is presented as follows:

$$\log\left(\frac{p_i}{1-p_i}\right) = -8.4166 + 0.0365(Glucose) + 0.0889(BMI) + 1.1243(SevenOrMorePregnancies)$$

$$\log\left(\frac{p_i}{1-p_i}\right) = -8.4166 + 0.0365(136) + 0.0889(31.2) + 1.1243(0)$$

$$p_{Glucose=136} = 0.3364$$

$$\log\left(\frac{p_i}{1-p_i}\right) = -8.4166 + 0.0365(137) + 0.0889(31.2) + 1.1243(0)$$

$$p_{Glucose=137} = 0.3446$$

Adding one unit (mg/dl) of `Glucose` raises the probability of diabetes by 0.3446 – 0.3364 = 0.0082. Applying the same approach to `BMI` and `SevenOrMorePregnancies`, an increase of one unit in `BMI` increases the probability by 0.0201. Additionally, experiencing seven or more pregnancies boosts the probability of developing diabetes by 0.2730.

References

Smith, J.W., Everhart, J., Dickson, W., Knowler, W. and 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.

Bilogur, Aleksey. (2023). missingno [Online]. Available at: https://github.com/ResidentMario/missingno (Accessed: 12 November 2023).

Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D. and Altman, R.B., 2001. Missing value estimation methods for DNA microarrays.

Appendix

Python Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.distributions.empirical_distribution import ECDF
from sklearn.impute import KNNImputer
from sklearn.linear model import LinearRegression
from sklearn.metrics import mean squared error, r2 score
import statsmodels.api as sm
import math
import missingno as msno
```

```
pima = pd.read_csv("PimaDiabetes.csv")
```

2. Exploratory Data Analysis

```
pima.head()
```

```
print(pima.isna().sum()) #checking null values
```

```
pimaNum = pima.drop('Outcome', axis=1)
```

```
pimaNum.describe()
```

```
pima['Outcome'].value_counts()
```

Univariate

```
#function to create histogram, kde, & ecdf for each feature
def univariateEDA(df):
  feature names = df.columns
  nVar = len(feature_names)
  fig, axs = plt.subplots(nVar,3, figsize=(20,25))
  for ind, col in enumerate(feature_names):
    for i in range(3):
      if i == 0:
        sns.histplot(df[col], kde=True, edgecolor='white', ax=axs[ind,i])
      elif i == 1:
        kdeAxes = sns.kdeplot(df[col], ax=axs[ind,i])
        kdeX, kdeY = kdeAxes.lines[0].get_data()
        kdeXmax = kdeX[np.argwhere(kdeY == kdeY.max())]
        kdeAxes.plot(df[col].mean()*np.ones(2), np.array([0, kdeY.max()]), '--b',
label='Mean')
        kdeAxes.plot(df[col].median()*np.ones(2), np.array([0, kdeY.max()]), '-.r', label =
'Median')
        kdeAxes.plot(kdeX[np.argwhere(kdeY == kdeY.max())].flatten()*np.ones(2), np.array([0,
kdeY.max()]), ':m', label='KDE-estimated Mode')
        kdeAxes.legend(fontsize='large')
      else:
        p25 = np.percentile(df[col].dropna(),25)
        p50 = np.percentile(df[col].dropna(),50)
        p75 = np.percentile(df[col].dropna(),75)
        p95 = np.percentile(df[col].dropna(),95)
        p99 = np.percentile(df[col].dropna(),99)

        ecdf = ECDF(df[col].dropna())
        ecdf.x.max()

        axs[ind,i].step(ecdf.x, 100*ecdf.y)
        axs[ind,i].plot([0,p25,p25],[25,25,0], '--k')
        axs[ind,i].plot([0,p50,p50],[50,50,0], '--k')
        axs[ind,i].plot([0,p75,p75],[75,75,0], '--k')
        axs[ind,i].plot([0,p95,p95],[95,95,0], '--k')
        axs[ind,i].plot([0,p99,p99],[99,99,0], '--k')
        axs[ind,i].set_xlabel(col)
        axs[ind,i].set_ylabel('ECDF x 100%')
        axs[ind,i].set_xlim(0, ecdf.x.max())
        axs[ind,i].set_ylim(0, 105)
        axs[ind,i].set xticks((0,p25,p50,p95,p99, ecdf.x.max()), (0,
        '$P {25}$','$P {50}$','$P {95}$','$P {99}$',ecdf.x.max()), fontsize=7)
```

```
  axs[0,0].set_title('Histogram', weight='bold')
  axs[0,1].set_title('KDE',weight='bold')
  axs[0,2].set_title('ECDF', weight='bold')

  plt.tight_layout()


  plt.show()
```

```
univariateEDA(pimaNum)
```

```
(pimaNum==0).sum(axis=0) #checking number of zero value
```

```
fig, axs = plt.subplots(2, 3, figsize=(15, 8))
features = ['Pregnancies','Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']

for i, feature in enumerate(features):
    row = i // 3  #integer division to determine row
    col = i % 3   #modulus to determine column
    sns.histplot(pimaNum[feature], ax=axs[row, col])

plt.show()
```

```
pimaMissing = pimaNum.replace(0, np.nan) #replace zero with NaN
```

```
pimaNumMissing = pimaNum.replace(0,np.nan)

msno.matrix(pimaNumMissing) #visualizing missing values pattern
plt.show()
```

Multivariate

```
n_cell = len(pimaNumMissing.columns)
plt.figure(figsize=(25,25))
for i in range(n_cell):
  for j in range(n_cell):
    plt.subplot(n_cell, n_cell, 1+i+(n_cell*j))
    if i == j:
      sns.kdeplot(pimaNumMissing.iloc[:,i], color='black')
      sns.histplot(pimaNumMissing.iloc[:,i], stat='density', color='lightseagreen')
    elif i<j:
      sns.kdeplot(x=pimaNumMissing.iloc[:,i], y=pimaNumMissing.iloc[:,j], cmap='Blues')
    else:
      sns.scatterplot(x=pimaNumMissing.iloc[:,i],y=pimaNumMissing.iloc[:,j], color='navy')

plt.tight_layout()
plt.show()
```

```
pimaNew = pd.concat([pimaNumMissing,pima['Outcome']], axis=1) #concat with the Outcome column
```

```
plt.figure(figsize=(8,8))
sns.heatmap(pimaNew.corr(), annot=True, fmt=".2f", cmap='coolwarm', cbar=True) #create
correlation heatmap
plt.show()
```

3. Adding & Modelling *SevenOrMorePregnancies*

```
pimaNew["SevenOrMorePregnancies"] = (pimaNew["Pregnancies"]>= 7).astype(int) #create
SevenOrMorePregnancies feature
```

```
#applying kNN imputer to Glucose, BloodPressure, BMI
imputer = KNNImputer(n_neighbors=5)
imputedData = imputer.fit_transform(pimaNew[['Glucose', 'BloodPressure', 'BMI']])

imputedPima = pd.DataFrame(imputedData, columns=['Glucose', 'BloodPressure', 'BMI'],
index=pimaNew.index)
imputedPima = pd.concat([imputedPima, pimaNew[['Age', 'DiabetesPedigree',
'SevenOrMorePregnancies','Outcome']]], axis=1)
```

```
#modeling single variable (SevenOrMorePregnancies) logistic regression
logitModel = sm.Logit.from_formula('Outcome ~ SevenOrMorePregnancies', data=imputedPima).fit()

print(logitModel.summary2()) #using summary2 to show AIC value
```

```
odds6 = math.exp(logitModel.params[0] + logitModel.params[1] * 0)
prob6 = odds6 / (1 + odds6)
print(f'Probability of developing diabetes, given six or fewer pregnancies is {prob6:.2f}')
```

```
odds7 = math.exp(logitModel.params[0] + logitModel.params[1] * 1)
prob7 = odds7 / (1 + odds7)
print(f'Probability of developing diabetes, given seven or more pregnancies is {prob7:.2f}')
```

4. Modelling with The Other Features

```
imputedPima.drop(['BloodPressure', 'DiabetesPedigree'], axis=1, inplace=True)
```

```
#create function to run every combination of variable using itertools library

from itertools import combinations

def fit_all_combinations(df, response_var):

  all_vars = list(df.columns)
  all vars.remove(response var)

  model_results = {}

  for i in range(1, len(all_vars) + 1):
    for combo in combinations(all vars, i):
      formula = "{} ~ {}".format(response var, ' + '.join(combo))
      model = sm.Logit.from_formula(formula, data=df).fit()
      model_results[formula] = model.aic
      print(formula)
      print(model.aic)

  return model_results
```

```
models = fit_all_combinations(imputedPima, "Outcome")
```

```
chosenModel = sm.Logit.from formula('Outcome ~ Glucose + BMI + SevenOrMorePregnancies', data =
imputedPima).fit()
print(chosenModel.summary2())
```

```
toPredict = pd.read_csv('ToPredict.csv') #read toPredict.csv
toPredict['SevenOrMorePregnancies'] = (toPredict['Pregnancies']>=7).astype(int) #add
SevenOrMorePregnancies column
```

```
#predicting diabetes from toPredict observations
probPrediction = chosenModel.predict(toPredict)

threshold = 0.5 #default threshold
toPredict['DiabetesPrediction'] = (probPrediction>=threshold).astype(int)
toPredict
```

```
#checking contribution of Glucose to probability
test = toPredict.iloc[0:1].copy() #using 1st row of toPredict
test.loc[1, :] = test.loc[0,:] #copy 1st row to 2nd row
test.loc[1, 'Glucose'] = 137 #replacing value to 1

probPrediction1 = chosenModel.predict(test.iloc[0])
probPrediction2 = chosenModel.predict(test.iloc[1])

print(f'Adding 1 unit of Glucose will increase probability by {probPrediction2[1] -
probPrediction1[0]:.4f}')
```

```
#checking contribution of BMI to probability
test = toPredict.iloc[0:1].copy() #using 1st row of toPredict
test.loc[1, :] = test.loc[0,:] #copy 1st row to 2nd row
test.loc[1, 'BMI'] = 32.2 #replacing value to 1

probPrediction1 = chosenModel.predict(test.iloc[0])
probPrediction2 = chosenModel.predict(test.iloc[1])

print(f'Adding 1 unit of BMI will increase probability by {probPrediction2[1] -
probPrediction1[0]:.4f}')
```

```
#checking contribution of SevenOrMorePregnancies to probability
test = toPredict.iloc[0:1].copy() #using 1st row of toPredict
test.loc[1, :] = test.loc[0,:] #copy 1st row to 2nd row
test.loc[1, 'SevenOrMorePregnancies'] = 1 #replacing value to 1
```

```
probPrediction1 = chosenModel.predict(test.iloc[0])
probPrediction2 = chosenModel.predict(test.iloc[1])

print(f'Having seven or more pregnancies will increase probability by {probPrediction2[1] -
probPrediction1[0]:.4f}')
```