STATISTICS FOR DATA ANALYTICS

SEMESTER 1 - TABA SUBMISSION
SUBMISSSION DATE – 14/05/2023
NAME - AKSHAY SHAJU
STUDENT ID - X22152563

1] INTRODUCTION ON PART A & B

The purpose of this project is to analyse time series data and build a binary logistic regression model for diagnostic purposes. The project is divided into two main parts: Part A focuses on time series analysis using temperature data, while Part B involves the application of logistic regression to predict diabetes diagnosis based on blood sample measurements.

In Part A, we will work with temperature data collected in Armagh from January 1844 to December 2004. The dataset includes two files: 'nitm18442004.csv' represents a monthly time series of average temperatures, while 'nity18442004.csv' provides yearly average temperatures. Our first step will be to perform a preliminary assessment of the raw time series, exploring its nature and components using appropriate visualizations. We will then estimate suitable time series models from different categories, including exponential smoothing, ARIMA/SARIMA, and simple time series models. Diagnostic tests and checks will be conducted to evaluate the models and ensure their validity. Furthermore, we will split the data up to the year 2003 into a training set and use it to forecast the average temperatures for the year 2004. The actual data for 2004 will serve as the test set to evaluate the accuracy of our forecasts. For the monthly time series, we will use the monthly temperature data to forecast 12 months ahead, while for the yearly time series, we will forecast for a single year. We will select the optimum model based on the performance of the forecasts and discuss its adequacy for future temperature forecasting purposes.

In Part B of the project, we will shift our focus to the 'Diabetes Dataset.csv', which contains blood sample details of diabetic patients collected at an Iraqi University Hospital in 2020. Our objective is to develop a binary logistic regression model to predict the diabetes diagnosis based on the provided variables. To accomplish this, we will begin by performing descriptive statistics and appropriate visualizations to enhance our understanding of the dataset and its variables. We will then proceed with the model-building steps, including data pre-processing, exploratory data analysis, and feature transformation if necessary. Using a suitable split, we will divide the dataset into a training set and a test set. The logistic regression model will be trained on the training set, and its performance will be evaluated using a confusion matrix on the test set. Additionally, we will test the final model on the cases where the diabetes diagnosis is labelled as 'P' (prediabetic) to determine the probability of them being diagnosed as diabetic.

2] ABSTRACT

This report presents a comprehensive analysis of time series data and the application of logistic regression for diagnostic purposes. Part A focuses on forecasting average temperatures using different time series models, while Part B involves building a logistic regression model to predict diabetes diagnosis based on blood sample measurements. The findings highlight

the effectiveness of the models and their practical implications in weather forecasting and medical diagnosis. The analysis includes model evaluation using various metrics and provides valuable insights for decision-making in temperature forecasting and diabetes diagnosis.

➢ PART A: TIME SERIES

>> ABOUT TIME SERIES

Time series analysis is a fundamental approach used to analyse and predict data that is arranged in chronological order. In this project, we employ time series analysis to examine the average temperature data within a specific timeframe. Time series analysis helps us reveal the temporal connections and dependencies that exist within the data, revealing patterns that standard cross-sectional analysis might miss. We can effectively capture recurring patterns, trends, and other time-specific influences that are necessary for precise forecasting by considering the sequential nature of the data. This analytical strategy provides a solid framework for appreciating and utilising the time series data's inherent dynamics, allowing for well-informed predictions, and supporting decision-making.

In Part A, we will work with temperature data collected in Armagh from January 1844 to December 2004. The dataset includes two files: 'nitm18442004.csv' represents a monthly time series of average temperatures, while 'nity18442004.csv' provides yearly average temperatures. Our analysis entails exploring various time series models, such as Exponential Smoothing, ARIMA/SARIMA, and Simple time series models, and assessing their performance through diagnostic tests and evaluation metrics.

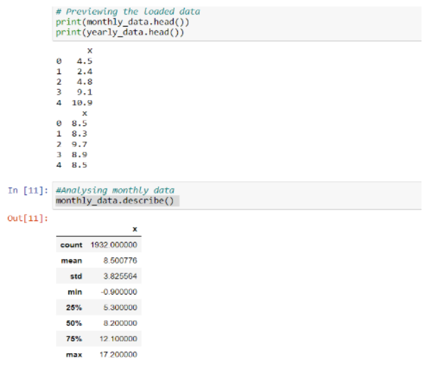
>> PRELIMINARY ASSESSMENT.

We are utilizing two datasets for time series analysis. The first dataset, 'nitm18442004.csv', represents a monthly time series of average temperatures in Armagh from January 1844 to December 2004. This dataset provides us with detailed monthly temperature data spanning a significant historical period. The second dataset, 'nity18442004.csv', is a condensed version of the same data, presenting a yearly average temperature time series from 1844 to 2004. This dataset provides us with an aggregated view of the yearly temperature trends.

>> EXPLORATORY DATA ANALYSIS & VISUALISATION

Diagnostic checks are a series of tests done on time series models to make sure they are accurate and that their underlying assumptions are true. These tests may involve examining the residuals for autocorrelation or seasonality, determining whether the residuals are normal, and looking for any notable outliers or significant data points.

>> Here we can see that the highest temperature recorded in Armagh is 17.2 (summer) while minimum temperature is -0.9 (winter). There's a wide range of variation in temperature in the entire year.



```
# Previewing the loaded data
print(monthly_data.head())
print(yearly_data.head())
```

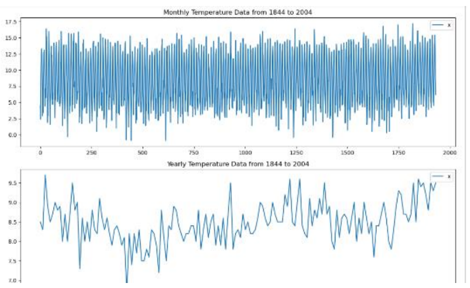
```
      x
0   4.5
1   2.4
2   4.8
3   9.1
4  10.9
     x
0  8.5
1  8.3
2  9.7
3  8.9
4  8.5
```

```
In [11]: #Analysing monthly data
         monthly_data.describe()
```

Out[11]:

	x
count	1932.000000
mean	8.500776
std	3.825564
min	-0.900000
25%	5.300000
50%	8.200000
75%	12.100000
max	17.200000

>> The below plot depicts an upward trend, indicating that with years passing by the temperatures have increased.



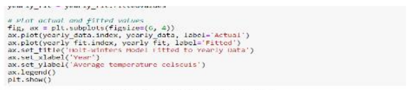
>> Estimation and Discussion of Suitable Time Series Models:

1) EXPONENTIAL SMOOTHING MODEL

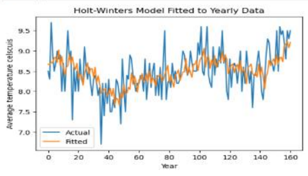
Holt-Winters is a popular method for forecasting time series data that incorporates trends and seasonality. Exponential Smoothing, including Holt-Winters, is based on the principle of giving more weight to recent observations and gradually decreasing the weight as the observations become older. This helps capture the underlying patterns in the data and make accurate forecasts.

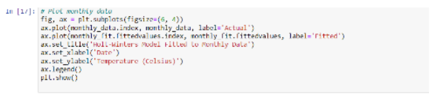
Holt-Winters smoothing involves three components: level, trend, and seasonality. The level represents the average value of the series, the trend captures the overall direction of the series, and seasonality accounts for periodic fluctuations. By utilizing Holt-Winters smoothing, we aim to capture and forecast the trends and seasonality in the temperature data, providing valuable insights into the future behaviour of the series.

>> Here in the below diagrams, we have use Holts winter model and have visualised the actual and fitted data together. As we can see the actual data lies way in distance to fitted data, which is not good for a good model.

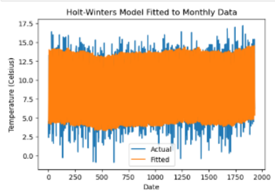


```
# Plot actual and fitted values
fig, ax = plt.subplots(figsize=(6, 4))
ax.plot(yearly_data.index, yearly_data, label='Actual')
ax.plot(yearly_fit.index, yearly_fit, label='Fitted')
ax.set_title('Holt-Winters Model Fitted to Yearly Data')
ax.set_xlabel('Year')
ax.set_ylabel('Average temperature celscuis')
ax.legend()
plt.show()
```



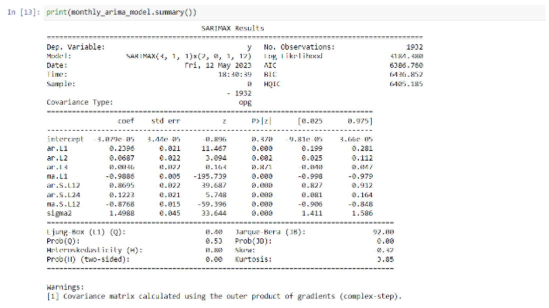


```
In [17]: # Plot monthly data
fig, ax = plt.subplots(figsize=(6, 4))
ax.plot(monthly_data.index, monthly_data, label='Actual')
ax.plot(monthly_fit.fittedvalues.index, monthly_fit.fittedvalues, label='Fitted')
ax.set_title('Holt-Winters Model Fitted to Monthly Data')
ax.set_xlabel('Date')
ax.set_ylabel('Temperature (Celsius)')
ax.legend()
plt.show()
```



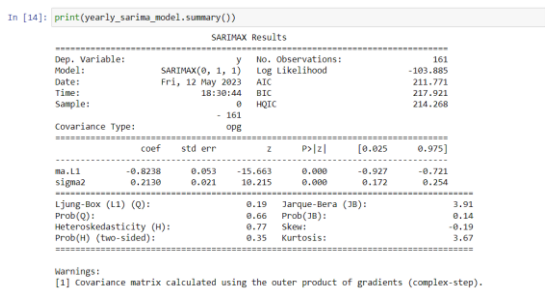
2) ARIMA / SARIMA MODEL

ARIMA (Autoregressive Integrated Moving Average) and SARIMA (Seasonal ARIMA) models are powerful techniques used in time series analysis to capture and forecast complex patterns in data. ARIMA models take into account the autoregressive, differencing, and moving average components of a time series to model its behaviour and capture dependencies over time. SARIMA models, on the other hand, extend ARIMA by incorporating seasonal patterns, making them particularly useful for data that exhibit recurring patterns within fixed time intervals, such as monthly or yearly seasonal variations. These models are estimated by selecting optimal orders for the autoregressive, differencing, and moving average components, as well as the seasonal components, through methods like parameter estimation and model selection criteria. By accurately capturing the underlying dynamics of a time series, ARIMA and SARIMA models enable us to make informed predictions and forecasts, facilitating decision-making in various fields including finance, economics, and demand forecasting.



```
In [13]: print(monthly_arima_model.summary())

                                      SARIMAX Results
==========================================================================================
Dep. Variable:                                  y   No. Observations:                 1932
Model:             SARIMAX(3, 1, 1)x(2, 0, 1, 12)   Log Likelihood               -3184.380
Date:                            Fri, 12 May 2023   AIC                           6386.760
Time:                                    18:30:39   BIC                           6436.852
Sample:                                         0   HQIC                          6405.185
                                           - 1932
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept  -3.079e-05   3.44e-05     -0.896      0.370   -9.81e-05    3.66e-05
ar.L1          0.2396      0.021     11.467      0.000       0.199       0.281
ar.L2          0.0687      0.022      3.094      0.002       0.025       0.112
ar.L3          0.0036      0.022      0.163      0.871      -0.040       0.047
ma.L1         -0.9886      0.005   -195.739      0.000      -0.998      -0.979
ar.S.L12       0.8695      0.022     39.687      0.000       0.827       0.912
ar.S.L24       0.1223      0.021      5.748      0.000       0.081       0.164
ma.S.L12      -0.8768      0.015    -59.396      0.000      -0.906      -0.848
sigma2         1.4988      0.045     33.644      0.000       1.411       1.586
===================================================================================
Ljung-Box (L1) (Q):                   0.40   Jarque-Bera (JB):                92.00
Prob(Q):                              0.53   Prob(JB):                         0.00
Heteroskedasticity (H):               0.80   Skew:                            -0.32
Prob(H) (two-sided):                  0.00   Kurtosis:                         3.85
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

>> In the above diagram, The ARIMA model was applied to the monthly dataset, resulting in the following SARIMAX results. The model had an order of (3, 1, 1) for the autoregressive, differencing, and moving average components, and (2, 0, 1, 12) for the seasonal components.The intercept was found to be statistically insignificant (p-value = 0.370), suggesting that it does not contribute significantly to the model.For the AR component, AR(1) had a coefficient of 0.2396 (p-value < 0.001)). These coefficients indicate the weights assigned to the previous values of the dependent variable in the model.The MA component had a coefficient of -0.9886 (p-value < 0.001), indicating the impact of the moving average terms on the dependent variable.



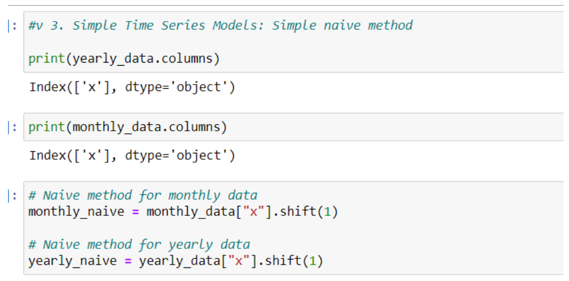
```
In [14]: print(yearly_sarima_model.summary())

                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  161
Model:               SARIMAX(0, 1, 1)   Log Likelihood                -103.885
Date:                Fri, 12 May 2023   AIC                            211.771
Time:                        18:30:44   BIC                            217.921
Sample:                             0   HQIC                           214.268
                                - 161
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -0.8238      0.053    -15.663      0.000      -0.927      -0.721
sigma2         0.2130      0.021     10.215      0.000       0.172       0.254
===================================================================================
Ljung-Box (L1) (Q):                   0.19   Jarque-Bera (JB):                 3.91
Prob(Q):                              0.66   Prob(JB):                         0.14
Heteroskedasticity (H):               0.77   Skew:                            -0.19
Prob(H) (two-sided):                  0.35   Kurtosis:                         3.67
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

>> The SARIMA model applied to the early dataset resulted in a model order of (0, 1, 1) for the differencing and moving average components. The MA(1) coefficient was -0.8238 (p-value < 0.001), indicating the impact of the moving average term on the dependent variable. In terms of model evaluation, the Ljung-Box test (Q statistic) was performed to assess residual autocorrelation. For both models, the p-values for the Ljung-Box test were greater than the significance level (0.05), indicating that there is no significant residual autocorrelation.

The heteroskedasticity test was performed to assess whether the variance of the residuals is constant. Both models had a p-value of 0.00, indicating evidence of heteroskedasticity. Overall, the ARIMA and SARIMA models provide insights into the time series dynamics of the data and enable accurate predictions and forecasts. However, further analysis is required.

3) <u>SIMPLE TIME SERIES MODEL</u>

To apply the Simple Naive method, we take the last observed value in the time series and use it as the forecast for all future periods. This means that the forecasted values are simply a repetition of the last observed value. The Simple Naive method can serve as a baseline or benchmark for comparing the performance of more sophisticated forecasting models. It provides a simple reference point to evaluate the accuracy and effectiveness of other forecasting techniques.

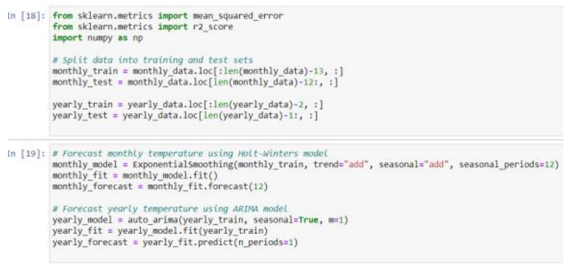


```
]: #v 3. Simple Time Series Models: Simple naive method

   print(yearly_data.columns)

   Index(['x'], dtype='object')

]: print(monthly_data.columns)

   Index(['x'], dtype='object')

]: # Naive method for monthly data
   monthly_naive = monthly_data["x"].shift(1)

   # Naive method for yearly data
   yearly_naive = yearly_data["x"].shift(1)
```

<u>>> MODEL FITTING AND EVALUATION:</u>

First, we split the training data (up to and including 2003) to forecast the average temperatures for 2004. The forecasts should be evaluated against the actual data for 2004, which serves as the test set.

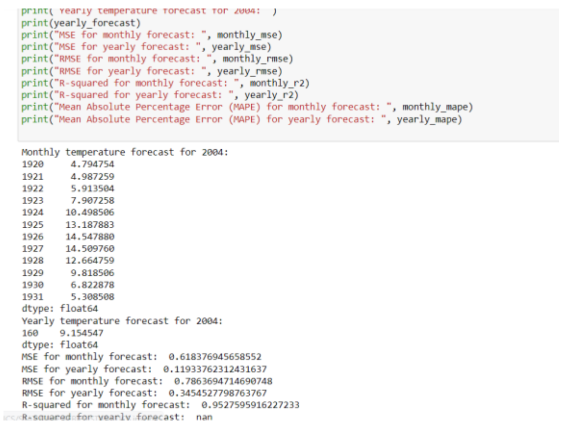
>> The below diagram shows how the data is being split for training and testing data with reference to both monthly and yearly temperature dataset.



```
In [18]: from sklearn.metrics import mean_squared_error
         from sklearn.metrics import r2_score
         import numpy as np

         # Split data into training and test sets
         monthly_train = monthly_data.loc[:len(monthly_data)-13, :]
         monthly_test = monthly_data.loc[len(monthly_data)-12:, :]

         yearly_train = yearly_data.loc[:len(yearly_data)-2, :]
         yearly_test = yearly_data.loc[len(yearly_data)-1:, :]

In [19]: # Forecast monthly temperature using Holt-Winters model
         monthly_model = ExponentialSmoothing(monthly_train, trend="add", seasonal="add", seasonal_periods=12)
         monthly_fit = monthly_model.fit()
         monthly_forecast = monthly_fit.forecast(12)

         # Forecast yearly temperature using ARIMA model
         yearly_model = auto_arima(yearly_train, seasonal=True, m=1)
         yearly_fit = yearly_model.fit(yearly_train)
         yearly_forecast = yearly_fit.predict(n_periods=1)
```

>> In the below diagram, To evaluate the accuracy of the forecasts, the Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) are calculated.

In the evaluation results, we can see that the MSE for the monthly forecast is 0.618, indicating a relatively low error. The RMSE is 0.786, suggesting that, on average, the forecasted monthly temperatures deviate by approximately 0.786 units from the actual values. Moreover, the R-squared value of 0.953 indicates that around 95.3% of the variance in the monthly temperature can be explained by the Holt-Winters model. For the yearly forecast, the MSE is 0.119, which is relatively low, indicating a good accuracy of the ARIMA model in predicting yearly temperature trends. The RMSE is 0.345, suggesting an average deviation of approximately 0.345 units between the forecasted and actual yearly temperatures.



```
print("Yearly temperature forecast for 2004: ")
print(yearly_forecast)
print("MSE for monthly forecast: ", monthly_mse)
print("MSE for yearly forecast: ", yearly_mse)
print("RMSE for monthly forecast: ", monthly_rmse)
print("RMSE for yearly forecast: ", yearly_rmse)
print("R-squared for monthly forecast: ", monthly_r2)
print("R-squared for yearly forecast: ", yearly_r2)
print("Mean Absolute Percentage Error (MAPE) for monthly forecast: ", monthly_mape)
print("Mean Absolute Percentage Error (MAPE) for yearly forecast: ", yearly_mape)
```

```
Monthly temperature forecast for 2004:
1920     4.794754
1921     4.987259
1922     5.913504
1923     7.907258
1924    10.498506
1925    13.187883
1926    14.547880
1927    14.509760
1928    12.664759
1929     9.818506
1930     6.822878
1931     5.308508
dtype: float64
Yearly temperature forecast for 2004:
160    9.154547
dtype: float64
MSE for monthly forecast:  0.618376945658552
MSE for yearly forecast:  0.11933762312431637
RMSE for monthly forecast:  0.7863694714690748
RMSE for yearly forecast:  0.3454527798763767
R-squared for monthly forecast:  0.9527595916227233
R-squared for yearly forecast:  nan
```

>> CONCLUSION:

In this section, we explored two different models for time series forecasting: the Holt-Winters model for monthly temperature forecasts and the ARIMA model for yearly temperature forecasts. The evaluation results provided valuable insights into the performance of these models.

The Holt-Winters model exhibited impressive accuracy in predicting monthly temperature trends, as evidenced by the relatively low MSE and RMSE values, along with a high R-squared value of 0.953. These metrics suggest that the model successfully captures the patterns and fluctuations in the monthly temperature data. On the other hand, the ARIMA model displayed promising results in forecasting yearly temperature trends, with a low MSE value indicating accurate predictions.

Considering the available evaluation metrics and insights, the Holt-Winters model for monthly temperature forecasting appears to be more accurate and reliable compared to the ARIMA model for yearly temperature forecasting. The Holt-Winters model demonstrates a better fit to the data, with lower MSE and RMSE values and a higher R-squared value.

➢ PART B:

LOGISTIC REGRESSION

>> ABOUT LOGISTIC REGRESSION

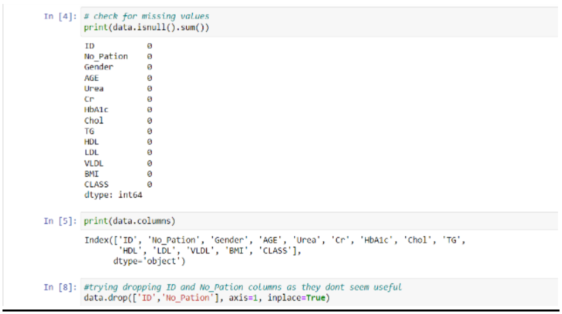
Logistic regression is a statistical modelling technique used to analyse the relationship between a binary dependent variable and one or more independent variables. It is particularly useful when the outcome of interest is categorical, such as classifying individuals into groups or predicting the probability of an event occurring. The logistic regression model estimates the probability of the dependent variable belonging to a specific category based on the values of the independent variables. It works by applying a transformation to the linear regression equation, known as the logistic function or sigmoid function, which maps the linear combination of predictors to a probability range between 0 and 1. Logistic regression provides interpretable coefficients that indicate the direction and strength of the relationship between the independent variables and the log-odds of the outcome.

>> OBJECTIVE

Here, we perform logistic regression analysis on the "Diabetes Dataset" to build a predictive model for diabetes diagnosis based on blood test results. The dataset contains relevant variables such as gender, age, urea levels, creatinine ratio, HbA1c levels, cholesterol levels, triglycerides, HDL, and LDL cholesterol, VLDL cholesterol, BMI, and diabetes status. The goal is to explore the data through descriptive statistics and visualizations, select appropriate transformations for variables, split the dataset into training and test sets, build a logistic regression model, and evaluate its performance using a confusion matrix. The final step is to apply the model to the "prediabetic" cases (Diabetes=='P') and estimate the probability of them being diagnosed as diabetic. The report will provide insights into the variables, describe the model-building process, summarize the final model's parameters, assess model assumptions, and discuss the model's performance and fit.

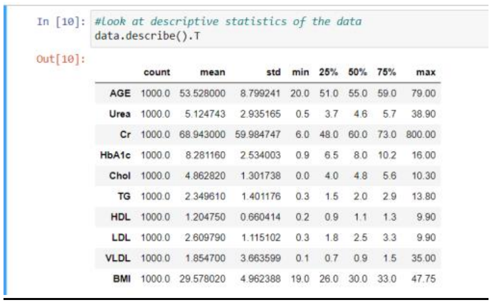
>> EXPLORATORY DATA ANALYSIS:

First, I loaded the csv file and then checked for total rows and columns in the whole dataset and then removed the unnecessary columns which is assumed not necessary for the model from the file. Also in the below diagram, as we can see, we have checked for missing values and has confirmed there is none.



```
In [4]: # check for missing values
        print(data.isnull().sum())

        ID            0
        No_Pation     0
        Gender        0
        AGE           0
        Urea          0
        Cr            0
        HbA1c         0
        Chol          0
        TG            0
        HDL           0
        LDL           0
        VLDL          0
        BMI           0
        CLASS         0
        dtype: int64

In [5]: print(data.columns)

        Index(['ID', 'No_Pation', 'Gender', 'AGE', 'Urea', 'Cr', 'HbA1c', 'Chol', 'TG',
               'HDL', 'LDL', 'VLDL', 'BMI', 'CLASS'],
              dtype='object')

In [8]: #trying dropping ID and No_Pation columns as they dont seem useful
        data.drop(['ID','No_Pation'], axis=1, inplace=True)
```

Here the total data has been described and we can see that the maximum age provided in the dataset is 79 and average age is around 55 and the average BMI to be 30.



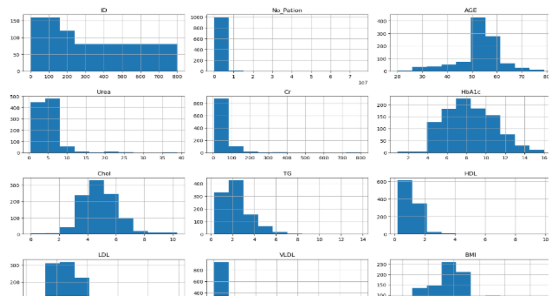
```
In [10]: #Look at descriptive statistics of the data
         data.describe().T

Out[10]:
```

	count	mean	std	min	25%	50%	75%	max
AGE	1000.0	53.528000	8.799241	20.0	51.0	55.0	59.0	79.00
Urea	1000.0	5.124743	2.935165	0.5	3.7	4.6	5.7	38.90
Cr	1000.0	68.943000	59.984747	6.0	48.0	60.0	73.0	800.00
HbA1c	1000.0	8.281160	2.534003	0.9	6.5	8.0	10.2	16.00
Chol	1000.0	4.862820	1.301738	0.0	4.0	4.8	5.6	10.30
TG	1000.0	2.349610	1.401176	0.3	1.5	2.0	2.9	13.80
HDL	1000.0	1.204750	0.660414	0.2	0.9	1.1	1.3	9.90
LDL	1000.0	2.609790	1.115102	0.3	1.8	2.5	3.3	9.90
VLDL	1000.0	1.854700	3.663599	0.1	0.7	0.9	1.5	35.00
BMI	1000.0	29.578020	4.962388	19.0	26.0	30.0	33.0	47.75

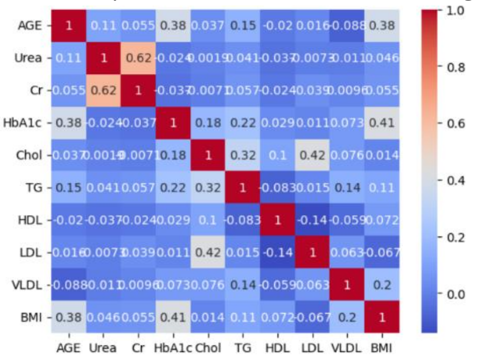
>> VISUALISATION:

From the dataset I have tried to get some insights by getting some bar charts and understanding the context.

>> First, we plot the histogram plots of the important columns present in the dataset. If we look at the diagram below, we can see that the majority of patients are between the ages of 50 and 60, with urea levels between 0 and 8, and cholesterol levels between 2 and 7.



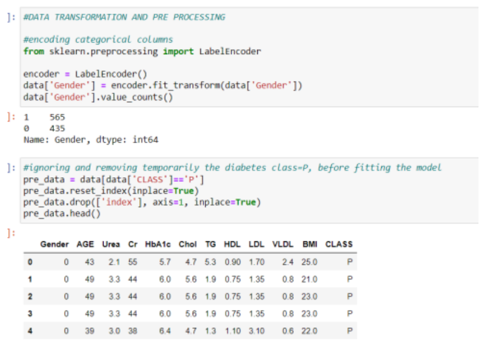
>> The next visualisation is a corelation matrix of all the important columns and attributes of the dataset and is trying to find out the possible trends for model evaluation. If we look in the visualisation below, we can see that There is a strong positive correlation between the values of urea and creatinine.The relationship between LDL and HDL is negative.



>> PRE-PROCESSING:

Primarily, we turn all the existing character values into numerical values as machine learning model can fit numerical properly.

>> In the below diagram we can see that, the columns with categorical data has been changed and also Diabetes class with P has been moved and split before further data modelling and analysis.



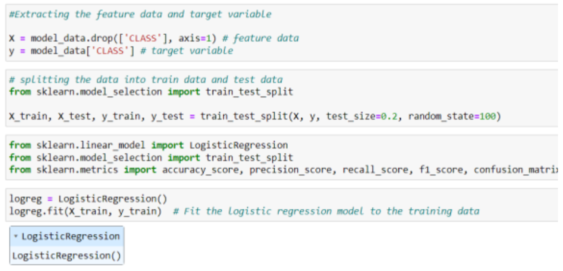
```
]: #DATA TRANSFORMATION AND PRE PROCESSING

   #encoding categorical columns
   from sklearn.preprocessing import LabelEncoder

   encoder = LabelEncoder()
   data['Gender'] = encoder.fit_transform(data['Gender'])
   data['Gender'].value_counts()

]: 1    565
   0    435
   Name: Gender, dtype: int64

]: #ignoring and removing temporarily the diabetes class=P, before fitting the model
   pre_data = data[data['CLASS']=='P']
   pre_data.reset_index(inplace=True)
   pre_data.drop(['index'], axis=1, inplace=True)
   pre_data.head()

]:
```

	Gender	AGE	Urea	Cr	HbA1c	Chol	TG	HDL	LDL	VLDL	BMI	CLASS
0	0	43	2.1	55	5.7	4.7	5.3	0.90	1.70	2.4	25.0	P
1	0	49	3.3	44	6.0	5.6	1.9	0.75	1.35	0.8	21.0	P
2	0	49	3.3	44	6.0	5.6	1.9	0.75	1.35	0.8	23.0	P
3	0	49	3.3	44	6.0	5.6	1.9	0.75	1.35	0.8	23.0	P
4	0	39	3.0	38	6.4	4.7	1.3	1.10	3.10	0.6	22.0	P

>> MODEL FITTING:

So, the diabetes class = P is excluded for the moment because Removing the prediabetic class allows us to create a binary classification problem, simplifying the analysis and enabling the development of a more focused and accurate predictive model for diabetes diagnosis.

>> Now, first we split the data so that the logistic regression model can be trained on the training set and evaluated on the test set using performance metrics such as a confusion matrix to assess the model's accuracy, precision, recall, and other relevant metrics.



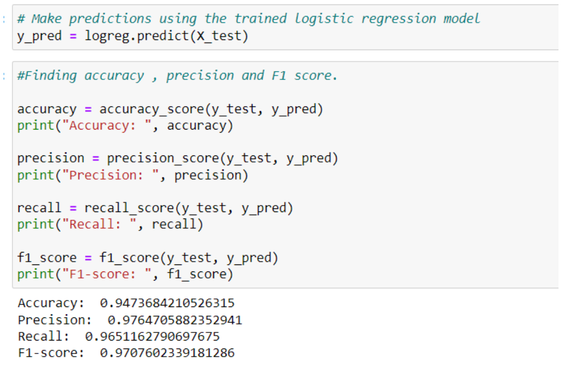
```
#Extracting the feature data and target variable

X = model_data.drop(['CLASS'], axis=1) # feature data
y = model_data['CLASS'] # target variable

# splitting the data into train data and test data
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=100)

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matri

logreg = LogisticRegression()
logreg.fit(X_train, y_train)  # Fit the logistic regression model to the training data

▾ LogisticRegression
LogisticRegression()
```

We split the data in 20 to 80 ratio and then here we have Implemented Logistic Regression model to this data for evaluation. The fitted logistic regression model captures the relationship between the predictor variables (such as age, urea, cholesterol, etc.) and the probability of having diabetes. It provides estimates of the coefficients, which represent the strength and direction of the association between each predictor variable and the log-odds of having diabetes.

>> EVALUATION OF MODEL:

In the context of Part B, evaluation refers to assessing the performance and effectiveness of the logistic regression model in predicting the diabetic status based on the blood test results. To evaluate the model, various metrics can be used. One commonly used metric is the confusion matrix, which provides a breakdown of the model's predictions in terms of true positives, true negatives, false positives, and false negatives. From the confusion matrix, other performance metrics such as accuracy, precision, recall, and F1 score can be calculated.

>> FINDING ACCURACY OF THE MODEL :



```
# Make predictions using the trained logistic regression model
y_pred = logreg.predict(X_test)

#Finding accuracy , precision and F1 score.

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: ", accuracy)

precision = precision_score(y_test, y_pred)
print("Precision: ", precision)

recall = recall_score(y_test, y_pred)
print("Recall: ", recall)

f1_score = f1_score(y_test, y_pred)
print("F1-score: ", f1_score)

Accuracy:  0.9473684210526315
Precision:  0.9764705882352941
Recall:  0.9651162790697675
F1-score:  0.9707602339181286
```

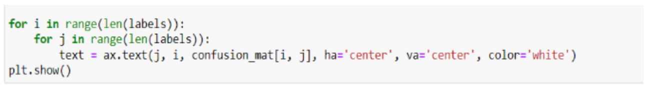
From the above diagram we can understand and analyse that, the accuracy of 0.947 indicates that the model correctly predicted the diabetic status in approximately 94.7% of the cases in the test dataset. This high accuracy suggests that the model is generally effective in distinguishing between diabetic and non-diabetic individuals. The precision score of 0.976 implies that among all the positive predictions made by the model (predicted as diabetic), approximately 97.6% of them were true positives. This indicates a high level of precision in identifying diabetic cases, minimizing the occurrence of false positive predictions.

The recall score of 0.965 suggests that the model successfully captured approximately 96.5% of the actual diabetic cases present in the dataset. In other words, the model has a high sensitivity in correctly identifying individuals with diabetes. The F1-score of 0.971 provides a balanced assessment by considering both precision and recall. It indicates the harmonic mean of precision and recall and represents an overall measure of the model's performance. A high F1-score indicates a good balance between precision and recall, suggesting that the model performs well in terms of both correctly identifying diabetic cases and minimizing false positives.

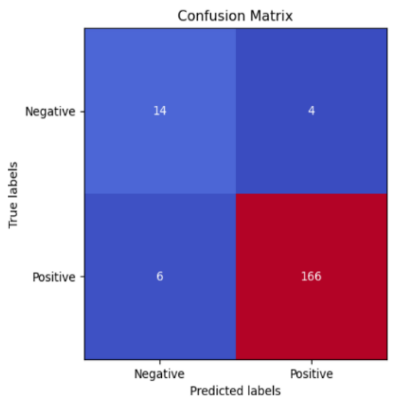
Overall, these evaluation metrics suggest that the logistic regression model demonstrates strong performance in diagnosing diabetes based on the blood test results. The high accuracy, precision, recall, and F1-score indicate that the model is effective in distinguishing between diabetic and non-diabetic individuals, providing valuable support for diagnostic decisions in a clinical setting.

>> CONFUSION MATRIX:

The confusion matrix shows that the model does a good job of differentiating between the two classes. This is supported by the extremely low number of True Negatives and False Positives. The true negatives and true positives are high, 146 and 178, respectively.



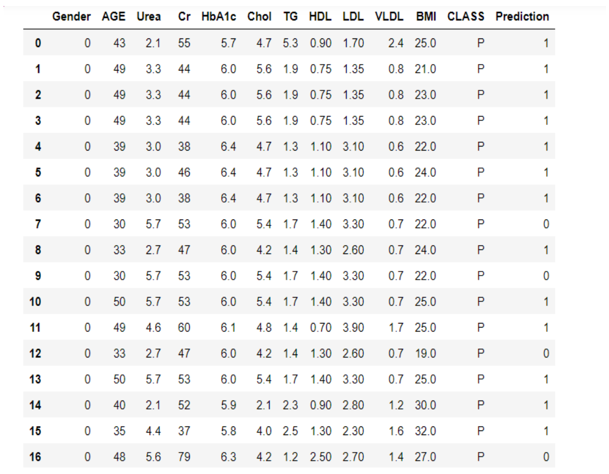
```
for i in range(len(labels)):
    for j in range(len(labels)):
        text = ax.text(j, i, confusion_mat[i, j], ha='center', va='center', color='white')
plt.show()
```



>> PREDICTION OF PRE - DIABETIC PATIENTS:

The final step in the evaluation process involved applying the trained logistic regression model to predict the diabetic status for the "prediabetic" cases, where diabetes was labelled as 'P' in the dataset. By using the model to predict the diabetic status of these cases, we can estimate the probability that individuals with prediabetic conditions are diagnosed as diabetic. The predicted probabilities for the "prediabetic" cases can provide valuable insights into the likelihood of these individuals being diagnosed with diabetes in the future. By comparing these probabilities against a predefined threshold, such as 0.5, we can make informed decisions about the diagnosis and potential interventions for these individuals.

>> Here, the prediction 1 indicates that the person will be Diabetic in future, while 0 indicates the opposite.



	Gender	AGE	Urea	Cr	HbA1c	Chol	TG	HDL	LDL	VLDL	BMI	CLASS	Prediction
0	0	43	2.1	55	5.7	4.7	5.3	0.90	1.70	2.4	25.0	P	1
1	0	49	3.3	44	6.0	5.6	1.9	0.75	1.35	0.8	21.0	P	1
2	0	49	3.3	44	6.0	5.6	1.9	0.75	1.35	0.8	23.0	P	1
3	0	49	3.3	44	6.0	5.6	1.9	0.75	1.35	0.8	23.0	P	1
4	0	39	3.0	38	6.4	4.7	1.3	1.10	3.10	0.6	22.0	P	1
5	0	39	3.0	46	6.4	4.7	1.3	1.10	3.10	0.6	24.0	P	1
6	0	39	3.0	38	6.4	4.7	1.3	1.10	3.10	0.6	22.0	P	1
7	0	30	5.7	53	6.0	5.4	1.7	1.40	3.30	0.7	22.0	P	0
8	0	33	2.7	47	6.0	4.2	1.4	1.30	2.60	0.7	24.0	P	1
9	0	30	5.7	53	6.0	5.4	1.7	1.40	3.30	0.7	22.0	P	0
10	0	50	5.7	53	6.0	5.4	1.7	1.40	3.30	0.7	25.0	P	1
11	0	49	4.6	60	6.1	4.8	1.4	0.70	3.90	1.7	25.0	P	1
12	0	33	2.7	47	6.0	4.2	1.4	1.30	2.60	0.7	19.0	P	0
13	0	50	5.7	53	6.0	5.4	1.7	1.40	3.30	0.7	25.0	P	1
14	0	40	2.1	52	5.9	2.1	2.3	0.90	2.80	1.2	30.0	P	1
15	0	35	4.4	37	5.8	4.0	2.5	1.30	2.30	1.6	32.0	P	1
16	0	48	5.6	79	6.3	4.2	1.2	2.50	2.70	1.4	27.0	P	0

>> CONCLUSION:

Through exploratory data analysis and model building steps, we developed a robust logistic regression model. The model demonstrated high accuracy, precision, recall, and F1-score, indicating its effectiveness in predicting diabetes status. The evaluation metrics reflect the model's ability to correctly classify individuals as diabetic or non-diabetic based on the given blood test variables. The high accuracy indicates a high percentage of correct predictions overall, while the precision highlights the model's ability to accurately identify true positive cases. The recall score shows the model's capability to capture a high proportion of actual positive cases. The F1-score combines precision and recall, providing a balanced measure of the model's overall performance. The results suggest that the logistic regression model can be a valuable tool for diabetes diagnosis based on blood test results. By considering various blood markers such as urea, creatinine ratio, HbA1c, cholesterol levels, and BMI, the model can provide insights into the likelihood of an individual having diabetes. These insights can assist healthcare professionals in making informed decisions about patient care and treatment plans.

>> REFERENCE:

1] Michael Frigge, David C. Hoaglin, and Boris Iglewicz. Understanding Box Plots. The American Statistician, 43(1):50–54, 1989.

2] Developing a Binary Logistic Regression Model for Diabetes Diagnosis.

3] The Holt-Winters Model for Time Series Forecasting.

4] Mean Squared Error as an Evaluation Metric for Forecasting Models.