DIABETES PREDICTION

DATA ANALYSIS PROJECT II

REPORT

Group 9

s15547 - Nisali Nuwandi

s15643 - Ranushi Nimthara

s15618 - Yashika Asiriwardhana

1. Abstract

Diabetes is a metabolic disorder characterized by the body's inability to properly regulate blood sugar levels. Left unchecked, it can lead to many complications that affect nearly every organ system in the body. But perhaps what's most alarming about diabetes is its subtle nature. It often goes undetected, silently chipping away at our health until it's too late. Here our analysis is focused on identifying the main factors that could impact the status of a person having diabetes or not and to develop an accurate predictive model that can identify the chances of a person having diabetes. Several models were considered and Random Forest was selected as the best model with 74.43% accuracy.

2. Table of content

List of Figures

List of Tables

3. Introduction

"The first wealth is health"

Diabetes is a significant global health challenge which is a chronic metabolic disorder. It is occurred by increased blood glucose levels. It is reported by the International Diabetes Federation(IDF), that approximately 537 million adults aged 20 to 80 were living with diabetes in 2021 and also this number is expected to rise 643 million by 2030. Diabetes is primarily classified into type1, type 2, and gestational diabetes. Where in type 1 diabetes it results from the autoimmune destruction of insulin-producing beta cells in the pancreas and occurring insulin deficiency. Type 2 diabetes develops due to insulin resistance and relative insulin deficiency and often linked with obesity. Gestational diabetes occurs during pregnancy and remain as risks to both mother and the fetal health.

Even though there are devices such as glucometer to directly measure the blood sugar of an individual, Lack of resources, high cost, lack of knowledge as well the difficulties to handle such devices most of the people don't have a direct method or resources to know their chances of having diabetes.

In response to these difficulties, our group objective is to come up with a statistical model that predict the chances of an individual to have diabetes. Our model is based on a comprehensive dataset, data cleaning, and analysis of these data. By solving the above-mentioned challenges, we try to address the complexity of determining whether a person is having diabetes as well provide them with most important factors that would cause them diabetes.

4. Description of the Problem

As diabetes is a huge health challenge faced by many people around the world, to identify the chances of having diabetes, what causes it and knowing how to prevent having diabetes by taking the right precautions is important. Early diagnosis can lead to lifestyle changes and more effective treatment, making predictive models for diabetes risk important tools for public and public health officials. Therefore we try understanding,

- What are the factors that influence a person to have diabetes?
- To come up with a suitable statistical model that can predict whether an individual has a chance of being diagnosed with diabetes.

5. Description of the Data Set

The dataset "Diabetes Health Indicators Dataset" was obtained by the Kaggle website, and the dataset contains a total of 70692 observations and 22 variables including the response variable. The response variable is the "Diabetes_binary" which is the status of having diabetes or not. This dataset is also based on USA.

Variable Name	Description
Quantitative Variables	
BMI	Body Mass Index of a person (BMI)
MentHlth	Number of days where the person's mental health was not good during the past 30 days.
PhysHlth	Number of days where the person's physical health was not good (an illness or an injury) during the past 30 days.
Qualitative Variables	
Diabetes_binary	Diabetes status of a person. (1-Have diabetes, 0-No diabetes)
HighBP	Whether have ever been told by a health professional that blood pressure is high (1-Yes, 0-No)
HighChol	Whether have ever been told by a health professional that blood cholesterol is high (1-Yes, 0-No)
CholCheck	Checked whether cholesterol within past 5 years. (1-Yes, 0-No
Smoker	Whether have ever smoked at least 100 cigarettes (1-Yes, 0-No)
Stroke	Have ever been told that had a stroke (1-Yes, 0-No)
HeartDiseasorAttack	Have ever reported having coronary heart disease or myocardial infarction (1-Yes, 0-No)
PhyActivity	Whether reported doing physical activity or exercise during past 30 days other than regular job. (1-Yes, 0-No)
Fruits	Consume Fruit 1 or more times per day (1-Yes, 0-No)
Veggies	Consume Vegetable 1 or more times per day (1-Yes, 0-No)
HvyAlcoholConsump	Whether a heavy drinker (1-Yes, 0-No)
AnyHealthcare	Whether have any kind of health care coverage, including insurance or prepaid plans (1-Yes, 0-No)
NoDocbCost	Whether there was a time in past 12 months when you needed to see a doctor but could not because of cost (1-Yes, 0-No)
GenHlth	General health of a person (1-Excellent, 2-Very good, 3-Good, 4-Fair, 5-Poor)
DiffWalk	Have any serious difficulty in walking or climbing stairs (1-Yes, 0-No)
Sex	Gender of the individual (1-Male, 0-Female)
Age	Age category of the individual (14 different categories)
Education	Highest grade or year of school the individual has completed (9 categories)

Table 1-Description of the variables

5.1 Data Preprocessing

Duplicates : There were no duplicate values in this dataset.

Missing values : There were also no missing values in the dataset.

Feature Engineering :

- There were 13 categories in the age variable which were :

1 : Age 18 - 24
2 : Age 25 - 29
3 : Age 30 - 34
4 : Age 35 - 39
5 : Age 40 - 44
6 : Age 45 - 49
7 : Age 50 - 54
8 : Age 55 - 59
9 : Age 60 - 64
10 : Age 65 - 69
11 : Age 70 - 74
12 : Age 75 - 79
13 : Age 80 or older

These 13 categories were recategorized into 4 categories as:
1 : 18 -29
2 : 30 - 49
3 : 50 - 69
4 : 70 or older

- There were 6 categories in Education variable, which were:

1 : Never attend school/ kindergarten
2 : Elementary
3 : Some high school
4 : High school graduate
5 : Some college or technical school
6 : College graduate

These 6 categories were recategorized into 4 categories as:

1. : Never attend school/ kindergarten
2. : High school
3. : Some college or technical school
4. : College graduate

- There were 8 categories in Income variable, which were:

1 : Less than $10,000

2 : $10,000 - $15,000

3 : $15,000 - $20,000

4 : $20,000 - $25,000

5 : $25,000 - $35,000

6 : $35,000 - $50,000

7 : $50,000 - $75,000

8 : $75,000 or more

These 8 categories were recategorized into 4 categories as:

1 : less than $25,000

2 : $25,000 - $50,000

3 : $50,000 - $75,000

4 : $75,000 or more

6. Descriptive Analysis

6.1. Univariate Analysis

Response Variable: Diabetes Status (Diabetes_binary)

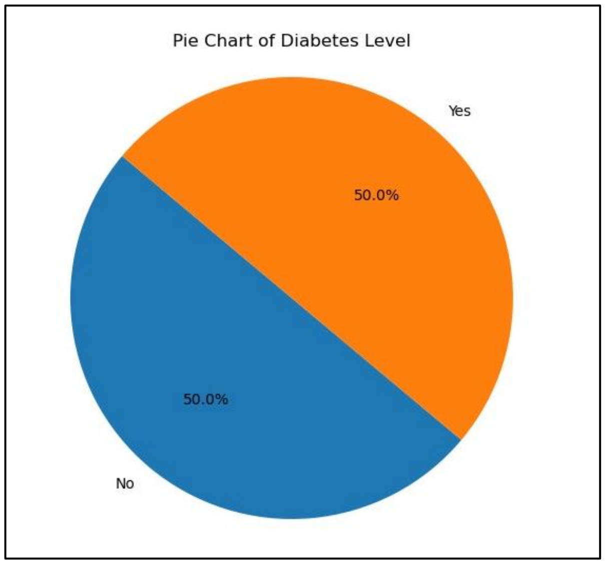


Figure 1-Pie chart of Diabetes status

The response variable of our analysis is the Diabetes_binary variable which is a binary variable which states the status of a person having diabetes.

It is to be seen that both of the two categories have 50% occurrences which implies that the data balancing is not needed.

No - Don’t have diabetes

Yes - Have Diabetes

Smoking Status

It can be seen that the percentage of individuals who smoke and who do not are approximately same with 47.3% and 52.7%.

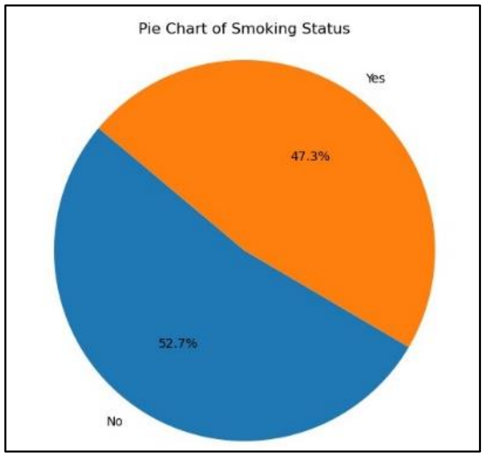


Figure 2-Pie chart of smoking status

HighBP

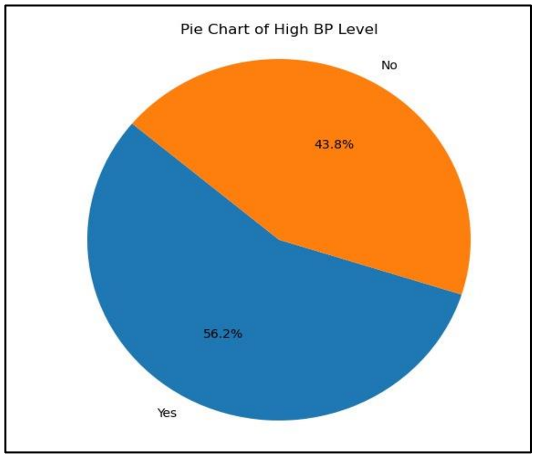


Figure 3-Pie chart of high blood pressure status

This variable indicates whether a person has been told by a health professional whether they have high blood pressure after testing. Percentage of people who had encountered high blood pressure was around 56.2%

BMI

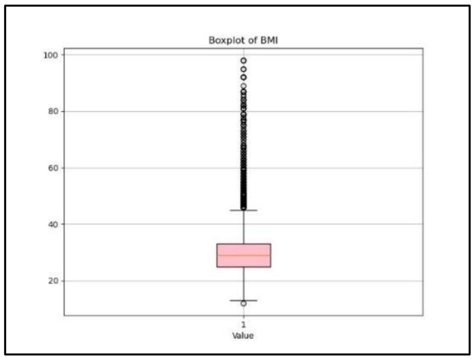


Figure 4-Boxplot of BMI

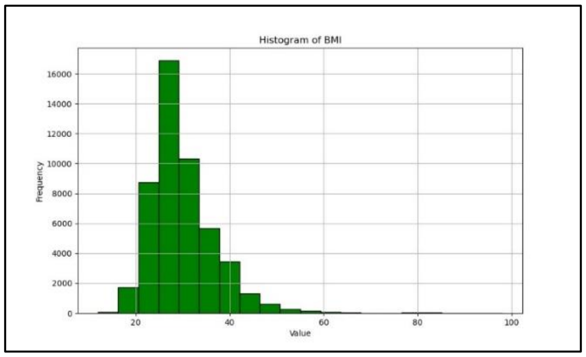


Figure 5-Histogram for BMI

It is to be seen that the histogram of the BMI variable is skewed to the right. Majority of the responses seems to be around 20 to 40. As we see from the boxplot, we can see that there are few outliers. We proceed to our analysis, with this in our mind.

General health

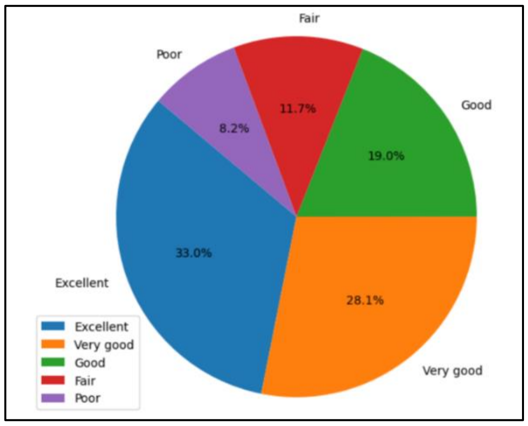


Figure 6-Pie chart of general health status

Majority of the people (33 %) thinks that they have an excellent general health and minority of the people thinks that their general health is poor.

Age

Majority (54.23%) of the people are in the 50-69 years age category. The smallest percentage belongs to the age category 18-29 years.

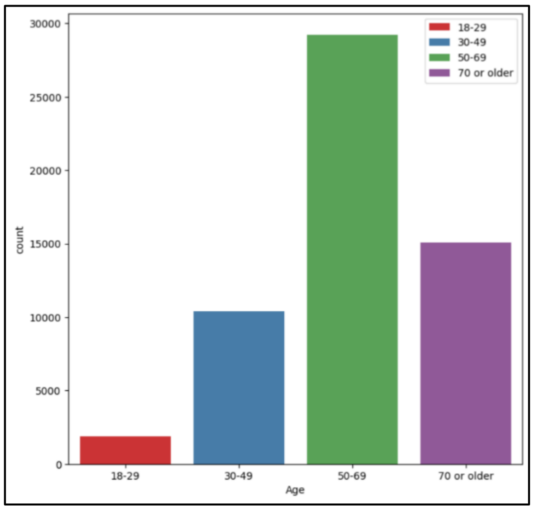


Figure 7-Bar chart of age categories

Income category

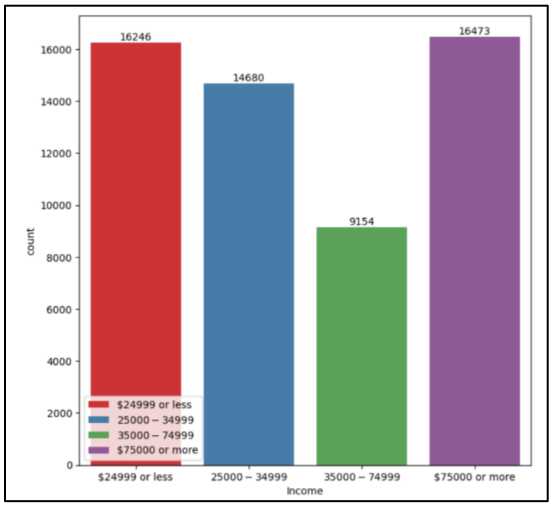


Figure 8-Bar chart of income category

Apparently the income categories "$24,999 or less" and "$75,000 or more" have similar number of percentages of people. Also, those income categories owns the majority of the people.

Heavy alcohol consumption

It is clearly visible that majority which is nearly 96% people here in our dataset are not heavy alcohol consumers. Only 4% of them are the heavy alcohol consumers.

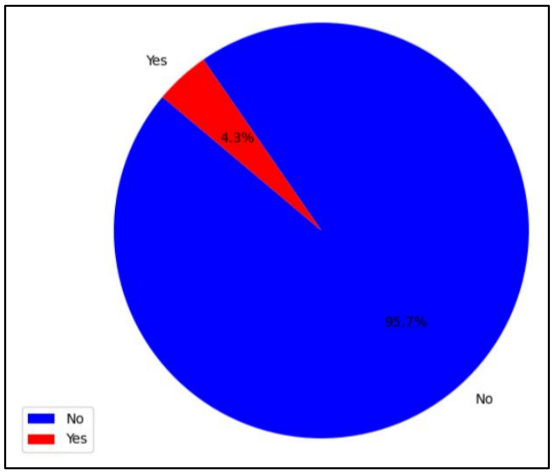


Figure 9-Pie chart of Heavy alcohol consumption

6.2.<u>Bivariate analysis</u>

HighBP vs Diabetes status

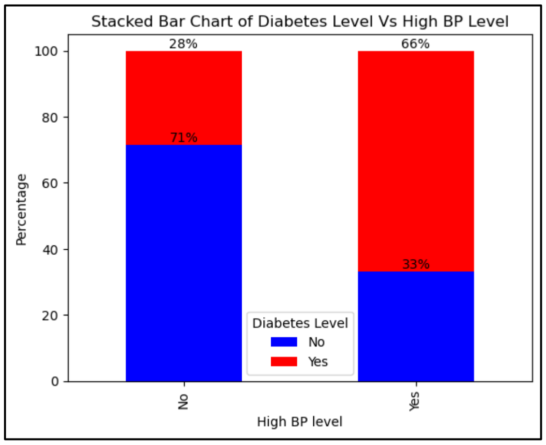


Figure 10-Stacked bar chart of HighBlood Pressure vs Diabetes status

It is clear to be seen that the High BP level and the Diabetes level have an association between them with the stacked bar chart shown here.

Percentage of people with high blood pressure who are having diabetes are around 66% where people with no high blood pressure having diabetes around 28%.

“High blood pressure is twice as likely to strike a person with diabetes as a person without diabetes.”

-Article on John Hopkins medicine-

Smoking status vs Diabetes status

It is to be seen that there's no significant difference in between diabetes status among smokers and not smokers as the percentages are near.

But contradictory to that, we found out a research paper with proof that says,

"Nicotine makes it harder to control your blood sugar"

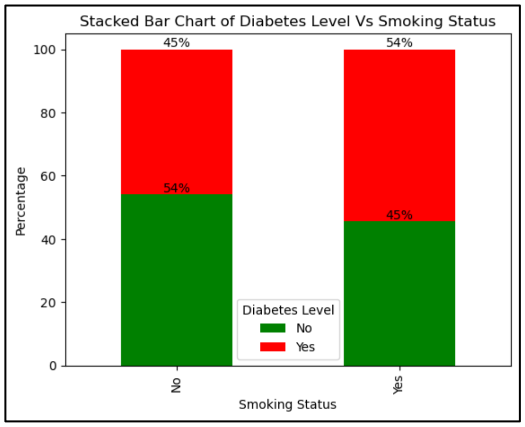


Figure 11-Stacked bar chart of smoking status vs Diabetes status

BMI vs Diabetes status

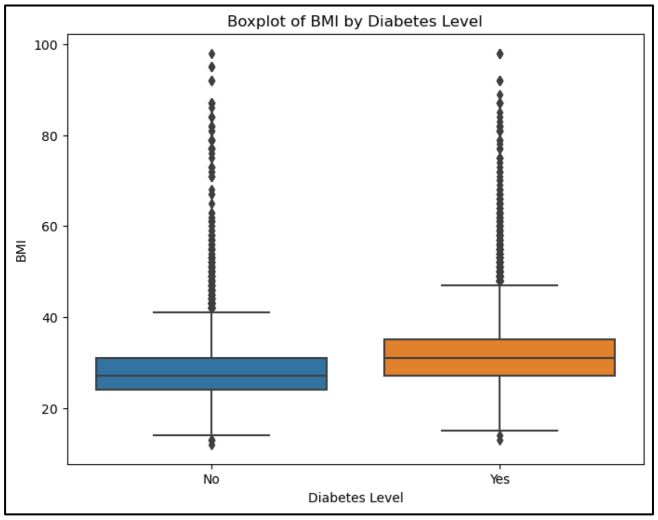


Figure 12-Boxplots of BMI vs Diabetes status

Median of the people who are having diabetes seems to be greater than the people with no diabetes. This indicates that there can be an association between diabetes status and the BMI of a person.

External research paper also confirmed it, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1890993/

"An increase in BMI is generally associated with a significant increase in prevalence of diabetes"

General health vs Diabetes Status

Here it is clear that as the stacked bar chart shows, when the general health is getting poorer, the percentage of people having diabetes is increasing.

In external resources are also confirming that as,

"Can reduce the risk of the diabetes by keeping blood pressure, blood glucose and cholesterol levels within recommended range."

-An article published on Better Health channel-

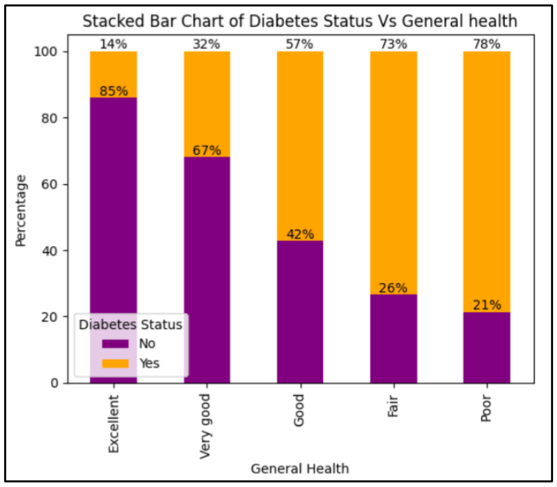


Figure 13-Stacked bar chart of general health vs Diabetes status

Age Vs Diabetes Status

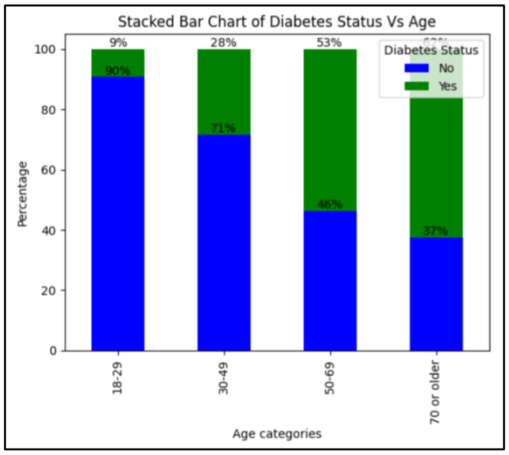


Figure 14-Stacked bar chart Age vs Diabetes status

Here in our dataset, it is clearly see that with the increase of the age the percentage of the people who have diabetes is increasing.

In external resources are also confirming that as,

"The elderly has a higher prevalence of diabetes and prediabetes than the young and middle-aged"

-A research on National library of Medicine USA -

Heavy alcohol consumption vs Diabetes status

Here in our dataset, it is visible that from the people who consume heavy alcohol amount, majority has no diabetes. But from the people who don't consume heavy alcohol amount majority has diabetes. In external resources we found that somewhat contradictory result as they say,

"Excess alcohol intake is associated with an increased risk of diabetes"

https://www.diabetes.org.uk/guide-to-diabetes/enjoy-food/what-to-drink-with-diabetes/alcohol-and-diabetes

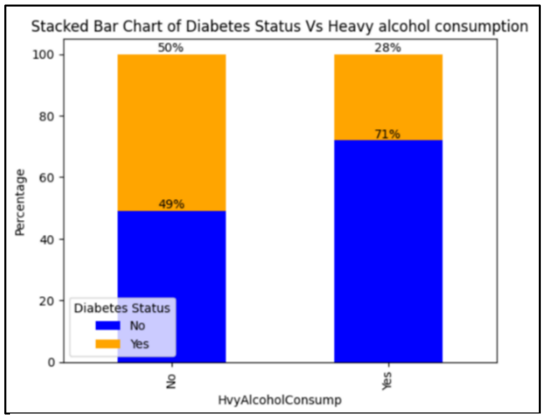


Figure 15-Stacked bar chart of HvyAlcoholconsump vs Diabetes status

6.3. Outlier Analysis

In outlier analysis, the Mahalanobis distance test for numerical variables was the main technique that we used. From that some relatively small number of outliers (97 outliers) were found.

And then also when the cluster analysis were done, by the score plot was identified that smaller number of outliers as well. But according to those numerical variable's definitions and external resources that was found, those outliers' corresponding variable values can be possible to have in the real world scenario. Therefore, those were not removed in the analysis.

6.4. Correlation among Variables

Correlation among the variables were checked using the gvif values.

Variables	VIF
High Bp	3.01
High Cholesterol	2.47
Cholesterol Check	32.13
BMI	18.19
Smoker	2.04
Stroke	1.16
Heart Disease or Attack	1.39
Physical Activity	3.82
Fruits	2.83
Veggies	5.19
High Alcohol Consumption	1.07
Any Healthcare	21.88
No Doctor Because of Cost	1.22
General Health	13.22
Mental Health	1.51
Physical Health	2.24
Difficulty to walk	2.09
Sex	1.99
Age	18.22
Education	16.22
Income	7.84

Table 2- VIF values of the variables

6.5.Factor Analysis for Mixed Data

To assess any relationship between the categorical variables in the dataset, factor analysis for mixed data is performed as the dataset consists with both numerical and categorical data.

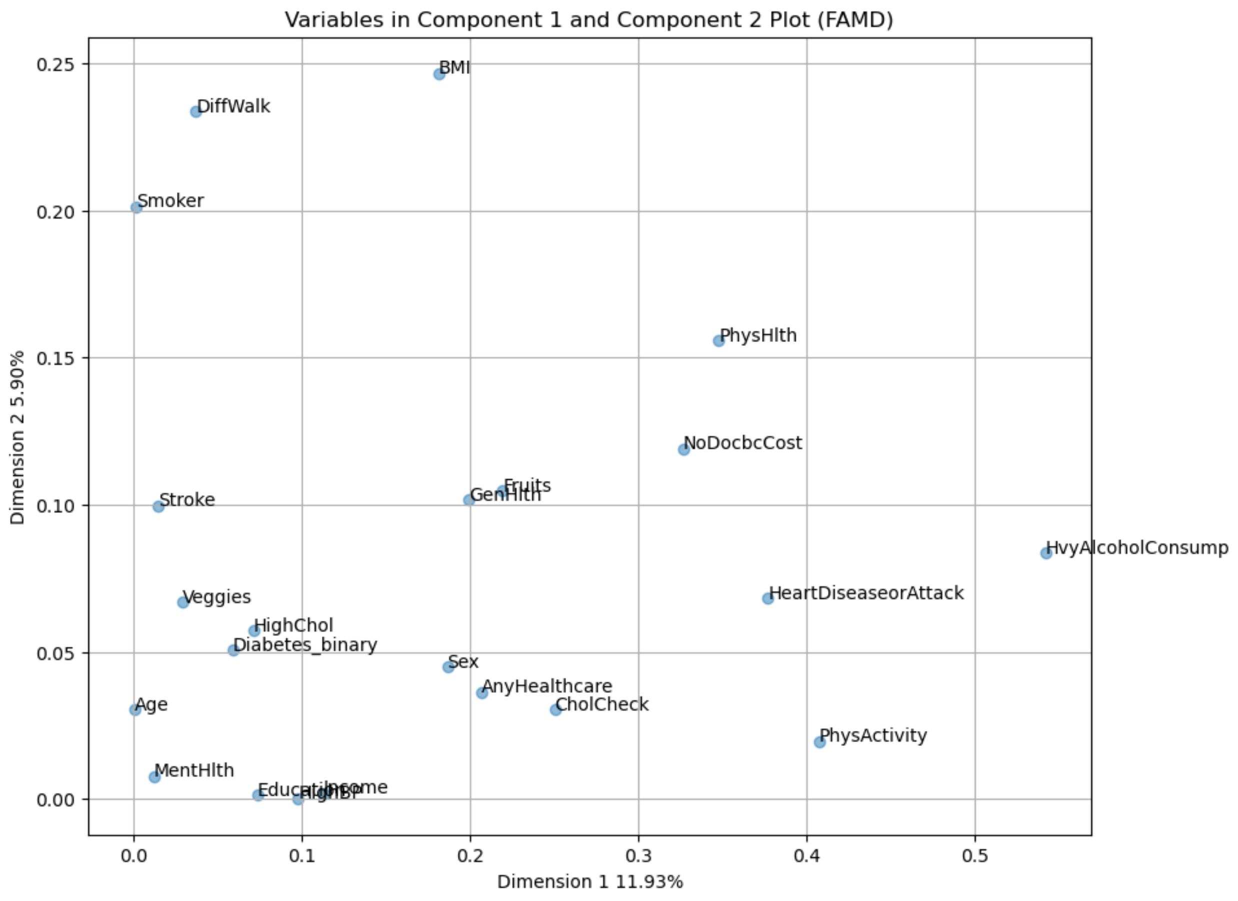


Figure 16-Loading plot of factor analysis for mixed data

It is seen from this score plot that the dimensions of the axis only explain 5.9% and 11.93%, which is very low comparatively. Therefore a better interpretation from this was not able. But to get an initial idea it can be seen that the variables such as BMI, High alcohol consumption, smoker, and Difficulty to walk have an association between the response variable. But it should be noted that these association can be wrong with the variability explained by the components are low.

6.6.Cluster Analysis

The cluster analysis was implemented to identify distinct groups or clusters of patients based on their health conditions and general life characteristics.

Checking whether clusters exist in the data

The elbow point is located at k=4 according to the elbow plot shown below.At that time, the silhouette width is measured, and it measures out to be 0.3865. It can be inferred that there is no quality clustering in the dataset as the value is less than 0.5.

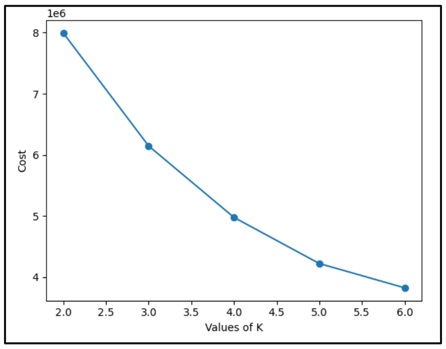


Figure 17-Elbow plot for K-prototype clustering

Final Finding of Descriptive Analysis:

- No clusters were found in the cluster analysis.
- There were some strong correlation between the variables such as BMI, High BP, GenHlth, Age with the response variable.
- Around 42 outliers were detected, but as those outliers seems to have a meaning they were not removed.
- All the variables were encoded in the dataset therefore we could proceed with the analysis without considering encoding.

Suggestions for Advanced Analysis:

- As the response variable, "Diabetes_binary" is a binary variable, logistic linear regression model was selected as the benchmark model for the advanced analysis.
- Accuracy which indicates how much was classified correctly from all the classifications, were selected as the model measure.

7. Advanced analysis

7.1. Logistic Linear Regression

With the response variable being binary, Logistic linear regression was selected as the benchmark model.

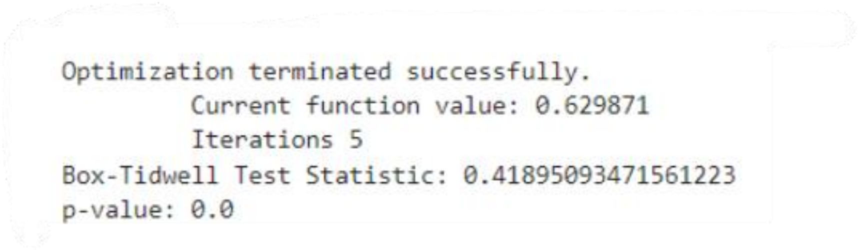
	Train Accuracy	**Test Accuracy**
Logistic Linear regression	0.7493	**0.7464**

Table 3-Accuracy measures of Logistic linear regression

Logistic regression model gives good accuracy, but we need to check for the assumption's validity as well.

Assumptions Validity:

- As previously it was found there are multi-collinearity between variables, the assumption of independence between independent variables are violated.
- As there were some outliers also that assumption was also violated.
- Checking for linearity between continuous variable and log odds :
  Using Box Tidwell test,



```
Optimization terminated successfully.
         Current function value: 0.629871
         Iterations 5
Box-Tidwell Test Statistic: 0.41895093471561223
p-value: 0.0
```

Figure 18-Results of Box-Tidwell test

As the p value is less than 0.05, Null hypothesis is rejected and linearity assumptions are violated.

7.2.<u>Ridge, Lasso & Elastic net regression</u>

It was found that there exists multi-collinearity in our dataset. Therefore, the regularization methods such as ridge, lasso and elastic net regression are applied. Since our target variable is a binary variable we here used Also, since we got 21 predictor variables in our dataset, it was needed a variable selection to do. Thus applying lasso and elastic net regression also satisfied that need as well. Accuracy measures of those ridge, lasso and elastic net regression can be get as follows. No overfitting is observed in all 3 regularization methods that we used.

	Train Accuracy	Test Accuracy
Ridge regression	0.748	0.7453
Lasso regression	0.726	0.721
Elastic net regression	0.749	0.746

Table 4-Accuracy measures of Ridge, Lasso, Elastic regressions

7.3.<u>KNN</u>

As KNN is a classification method that doesn't consider linearity assumptions, KNN was also fitted to our data.

	Train Accuracy	Test Accuracy
KNN	0.79	**0.71**

Table 5-Accuracy measures of KNN

Here k = 5 as the optimal model and as our response variable is a binary variable, "hamming distance" is selected as the distance measure. Here it is seen that the model consists of overfitting.

7.4. XG boost

XG boost is also a powerful ensemble machine learning technique which can use for classification and regression purposes. Since these ensemble methods are robust to the outliers and multicollinearity, this was applied to our dataset. Here some of the parameters were tuned using the grid search method in order to get a better accuracy,

Best parameters: {“n_estimators”:250, “learning_rate”: 0.3, “max depth”: 3}

	Train Accuracy	Test Accuracy
XG boost	0.7587	0.7463

Table 6-Accuracy measures of XG boost

7.5. Random Forest

The Random forest classifier was fitted for the dataset with the default parameters and then we came up with the following results.

	Train	Test
Random Forest	0.9853	0.7275

Table 7-Accuracy measures of Random forest with default parameters

Since there is overfitting in the random forest with default parameters, they were tuned and the results are given below.

Best parameters:{‘n_estimators’:300,min_samples_split’:4,’max_depth’:10}

	Train	Test
Random Forest	**0.7707**	**0.7475**

Table 8-Accuracy measures of Random forest after hyper parameter tuning

Now when the results are compared with the results of the other models it was decided that the random forest is the best model for predicting diabetes.

Then the most important variables for predicting diabetes were found by the below variable importance graph of the Random forest.

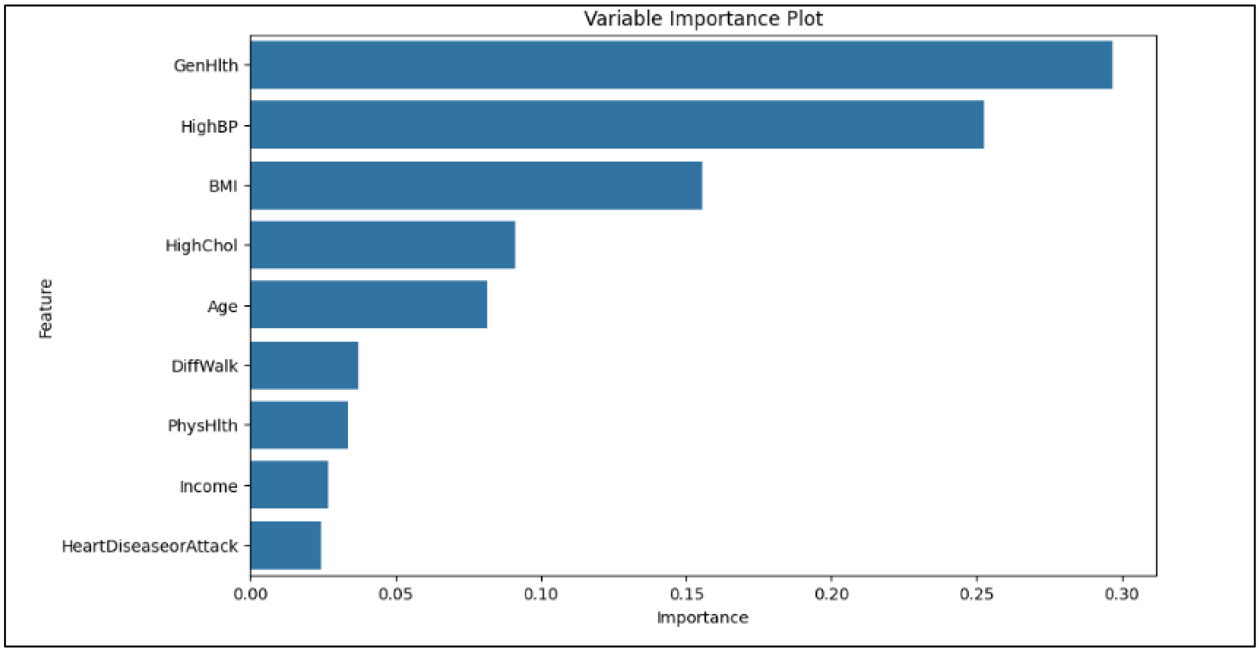


Figure 19-Variable Importance Plot of Random forest

Then random forest model is fitted for the most important variables and the results are given below.

	Train	Test
Random Forest	**0.7652**	**0.7443**

Table 9-Accuracy measures of Random forest with most important variables

Then it can be indicated that the accuracy of the model for the most important variables is almost the same as the model with all variables.

Partial dependency plots

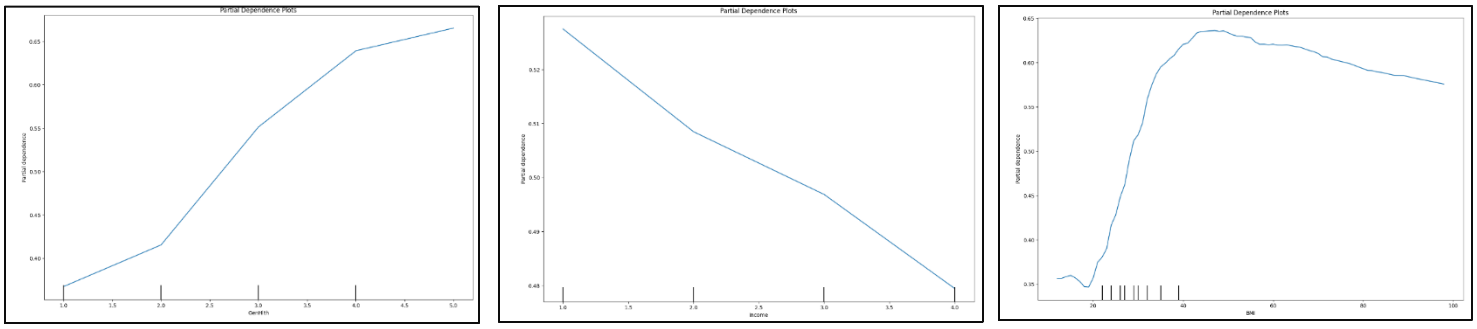


Figure 20-Partial Dependency plots

According to the partial dependency plots, the general health has an upward trend and it describes that when the general health becomes poor it is associated with a higher risk of diabetes whereas an increase in income is associated with a lower risk of diabetes.

When it comes to the BMI the risk of having diabetes is increased to some extent and then decreased to somewhat lower. But the difference is considerably high.

So it is indicated that the BMI is also an important variable in predicting diabetes.

8. Issues encountered and proposed solutions

- Since some categorical predictors (Education, Income, Age) had many factor levels, re-categorization of those variables was done to reduce the number of dummy variables obtained when fitting the models and to increase interpretability
- Computational limitations to handle large datasets: Time-consuming parameter adjustment and extremely long code execution durations were noted.

9. Discussion and Conclusion

- In our analysis, we identified some outliers by the Mahalanobis distance method. So, after we investigated those outliers and identified that those outliers have some meaning. Thus the outliers were not removed from the dataset.
- The dataset is balanced. Thus the accuracy is selected as the evaluation matric. So the results obtained by the Advanced Analysis can be summarized as below.

	Train Accuracy	Test Accuracy
Logistic Regression	**0.7493**	**0.7464**
Ridge Regression	**0.7484**	**0.7453**
Lasso Regression	**0.726**	**0.721**
Elastic Net Regression	**0.749**	**0.746**
Random Forest	**0.7707**	**0.7475**
XG Boost	**0.7587**	**0.7463**
KNN	**0.79**	**0.71**

Table 10-Comparison of the all models

- So according to the accuracy measurement, it can be visible that the Random Forest is the best model for predicting diabetes. Then we identified that “General health”,” BMI”,” High Cholesterol” and” Age” are mostly associated with diabetes.
- These results offer important new information about the variables influencing diabetes.
- Therefore, anyone can use these data to gain a better understanding of their diabetic condition, and healthcare professionals may also find these results useful in assessing the diabetes conditions of their patients.
- As a result, the model's target audience benefits as much as possible while the study's goals are met.

10. Appendix

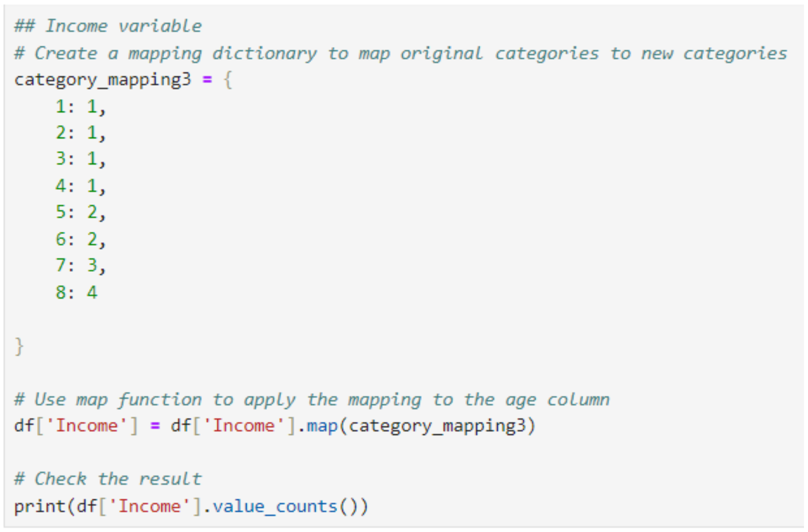
Float types covered into integer

```python
## all float types are coverted into integer
df = df.astype(int)
df['BMI'] = df['BMI'].astype(float)
df['MentHlth'] = df['MentHlth'].astype(float)
df['PhysHlth'] = df['PhysHlth'].astype(float)
```

Re-categorizing Variables

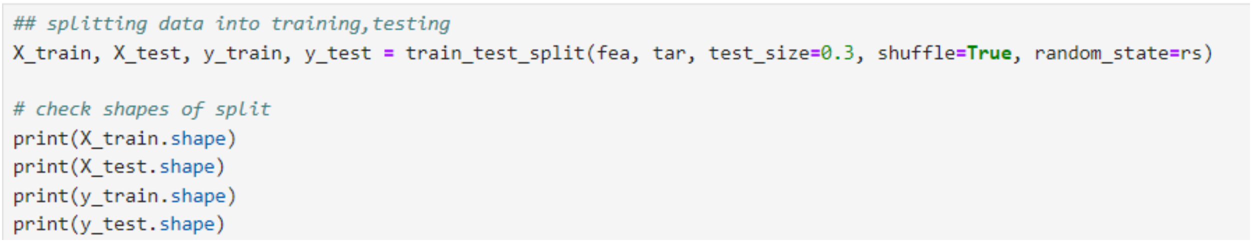
```python
## age variable
# Create a mapping dictionary to map original categories to new categories
category_mapping = {
    1: 1,
    2: 1,
    3: 2,
    4: 2,
    5: 2,
    6: 2,
    7: 3,
    8: 3,
    9: 3,
    10: 3,
    11: 4,
    12: 4,
    13: 4

}

# Use map function to apply the mapping to the age column
df['Age'] = df['Age'].map(category_mapping)

# Check the result
print(df['Age'].value_counts())
```

```python
## Education variable
# Create a mapping dictionary to map original categories to new categories
category_mapping2 = {
    1: 1,
    2: 2,
    3: 2,
    4: 2,
    5: 3,
    6: 4

}

# Use map function to apply the mapping to the age column
df['Education'] = df['Education'].map(category_mapping2)

# Check the result
print(df['Education'].value_counts())
```



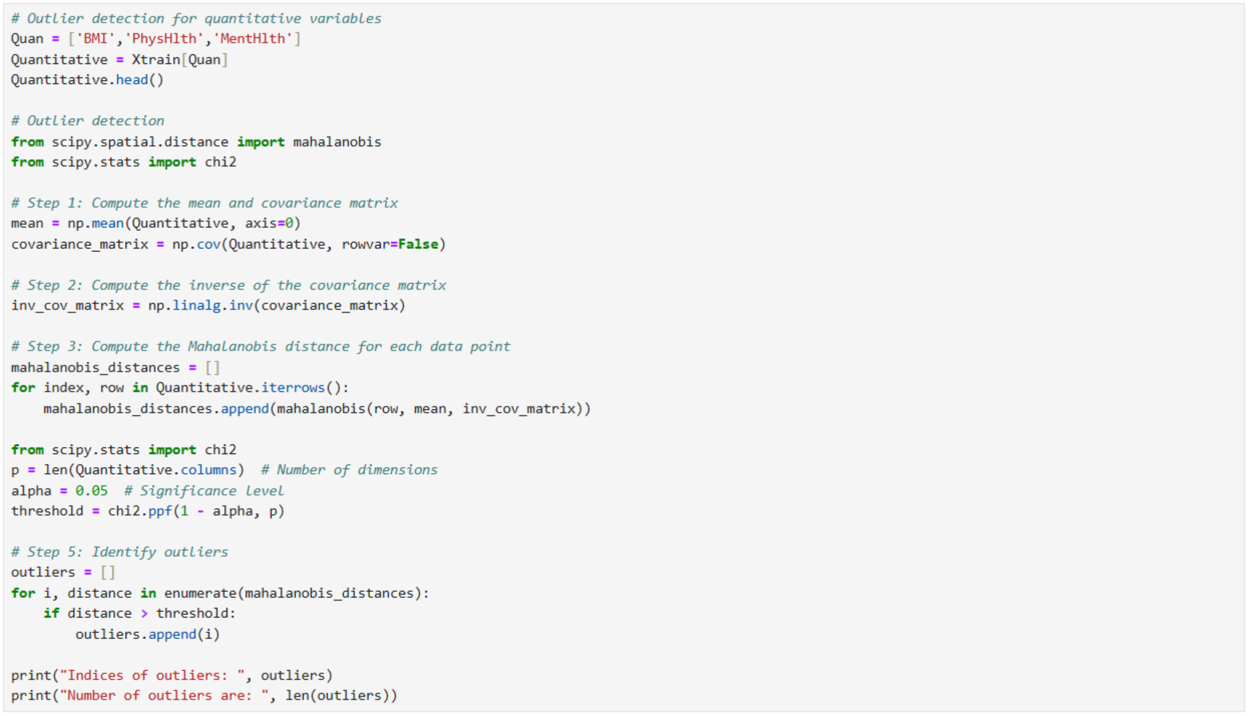
```
## Income variable
# Create a mapping dictionary to map original categories to new categories
category_mapping3 = {
    1: 1,
    2: 1,
    3: 1,
    4: 1,
    5: 2,
    6: 2,
    7: 3,
    8: 4

}

# Use map function to apply the mapping to the age column
df['Income'] = df['Income'].map(category_mapping3)

# Check the result
print(df['Income'].value_counts())
```

Split data into training testing



```
## splitting data into training,testing
X_train, X_test, y_train, y_test = train_test_split(fea, tar, test_size=0.3, shuffle=True, random_state=rs)

# check shapes of split
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

Outlier analysis



```
# Outlier detection for quantitative variables
Quan = ['BMI','PhysHlth','MentHlth']
Quantitative = Xtrain[Quan]
Quantitative.head()

# Outlier detection
from scipy.spatial.distance import mahalanobis
from scipy.stats import chi2

# Step 1: Compute the mean and covariance matrix
mean = np.mean(Quantitative, axis=0)
covariance_matrix = np.cov(Quantitative, rowvar=False)

# Step 2: Compute the inverse of the covariance matrix
inv_cov_matrix = np.linalg.inv(covariance_matrix)

# Step 3: Compute the Mahalanobis distance for each data point
mahalanobis_distances = []
for index, row in Quantitative.iterrows():
    mahalanobis_distances.append(mahalanobis(row, mean, inv_cov_matrix))

from scipy.stats import chi2
p = len(Quantitative.columns)  # Number of dimensions
alpha = 0.05  # Significance level
threshold = chi2.ppf(1 - alpha, p)

# Step 5: Identify outliers
outliers = []
for i, distance in enumerate(mahalanobis_distances):
    if distance > threshold:
        outliers.append(i)

print("Indices of outliers: ", outliers)
print("Number of outliers are: ", len(outliers))
```

Factor analysis for mixed data

```python
famd = FAMD(
    n_components=2,
    n_iter=3,
    copy=True,
    check_input=True,
    random_state=42,
    engine="sklearn",
    handle_unknown="error"  # same parameter as sklearn.preprocessing.OneHotEncoder
)
famd = famd.fit(df1)
```

```python
eig_sum = famd.eigenvalues_summary
eig_sum
```

```python
c = eig_sum['% of variance']
c
```

```python
import altair as alt
```

```python
alt.data_transformers.disable_max_rows()
```

```
DataTransformerRegistry.enable('default')
```

```python
famd.column_contributions_.style.format('{:.0%}')
```

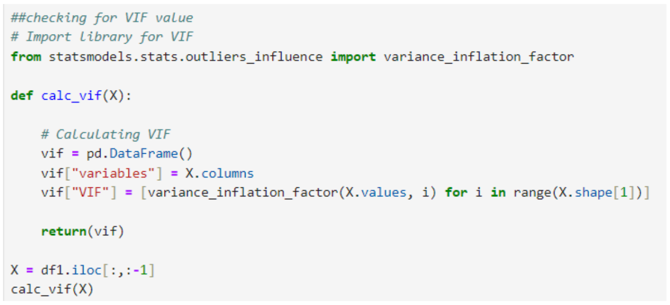
```python
# Convert col to a NumPy array
col_array = np.array(col)

# Convert the array to a DataFrame
col_df = pd.DataFrame(col_array, columns=['Component 1', 'Component 2'])
# Print the DataFrame
print(col_df)
```

```python
famd.plot(
    df1,
    x_component=0,
    y_component=1
)
```

```python
column_names = df1.columns
print(column_names)
```

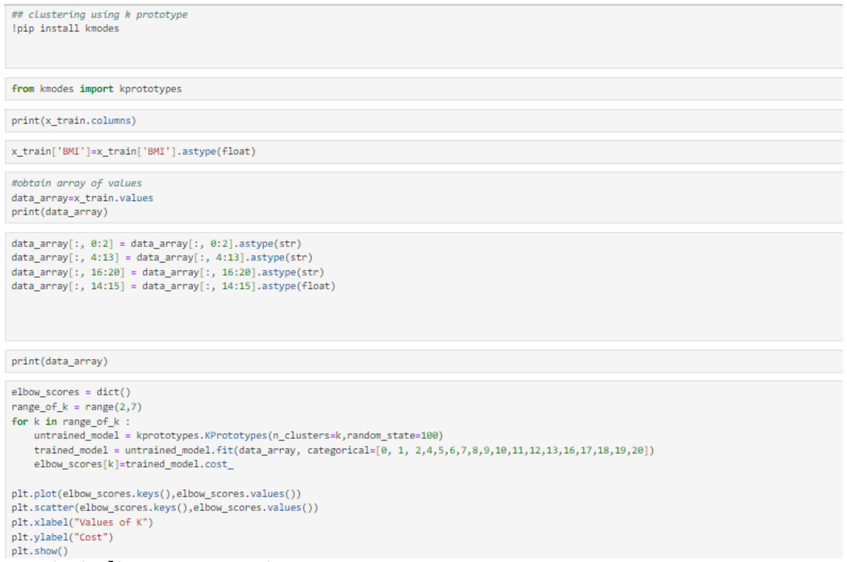
```python
# Plot the variables in the FAMD space
plt.figure(figsize=(10, 8))
plt.scatter(col[0], col[1], alpha=0.5)

# Add labels for each point (variable)
for i, txt in enumerate(df1.columns):
    plt.annotate(txt, (col[0][i], col[1][i]))

# Plot the component 1 and component 2 points
#plt.scatter(famd.row_coordinates_[0], famd.row_coordinates_[1], color='red', label='Component 1 and Component 2')

# Extract the percentages
percentage_dimension1 = c.loc[0]
percentage_dimension2 = c.loc[1]

plt.title('Variables in Component 1 and Component 2 Plot (FAMD)')
#xlabel = 'Dimension 1' + " " + percentage_dimension1
plt.xlabel('Dimension 1' + " " + percentage_dimension1)
plt.ylabel('Dimension 2' + " " + percentage_dimension2)


#plt.title('Variables in Component 1 and Component 2 Plot (FAMD)')
#plt.xlabel('Dimension 1')
#plt.ylabel('Dimension 2')
#plt.legend()
plt.grid(True)
plt.savefig('FAMD_Variables_Plot.png')
plt.show()
```

Checking VIF values



```
##checking for VIF value
# Import library for VIF
from statsmodels.stats.outliers_influence import variance_inflation_factor

def calc_vif(X):

    # Calculating VIF
    vif = pd.DataFrame()
    vif["variables"] = X.columns
    vif["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]

    return(vif)

X = df1.iloc[:,:-1]
calc_vif(X)
```

Cluster analysis



```
## clustering using k prototype
!pip install kmodes
```

```
from kmodes import kprototypes
```

```
print(x_train.columns)
```

```
x_train['BMI']=x_train['BMI'].astype(float)
```

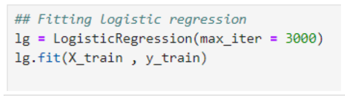
```
#obtain array of values
data_array=x_train.values
print(data_array)
```

```
data_array[:, 0:2] = data_array[:, 0:2].astype(str)
data_array[:, 4:13] = data_array[:, 4:13].astype(str)
data_array[:, 16:20] = data_array[:, 16:20].astype(str)
data_array[:, 14:15] = data_array[:, 14:15].astype(float)
```

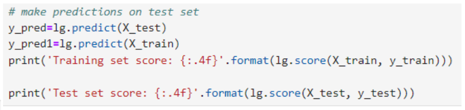
```
print(data_array)
```

```
elbow_scores = dict()
range_of_k = range(2,7)
for k in range_of_k :
    untrained_model = kprototypes.KPrototypes(n_clusters=k,random_state=100)
    trained_model = untrained_model.fit(data_array, categorical=[0, 1, 2,4,5,6,7,8,9,10,11,12,13,16,17,18,19,20])
    elbow_scores[k]=trained_model.cost_

plt.plot(elbow_scores.keys(),elbow_scores.values())
plt.scatter(elbow_scores.keys(),elbow_scores.values())
plt.xlabel("Values of K")
plt.ylabel("Cost")
plt.show()
```

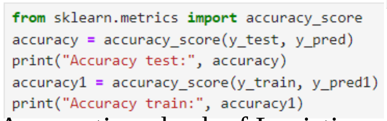
Logistic linear regression



```
## Fitting logistic regression
lg = LogisticRegression(max_iter = 3000)
lg.fit(X_train , y_train)
```

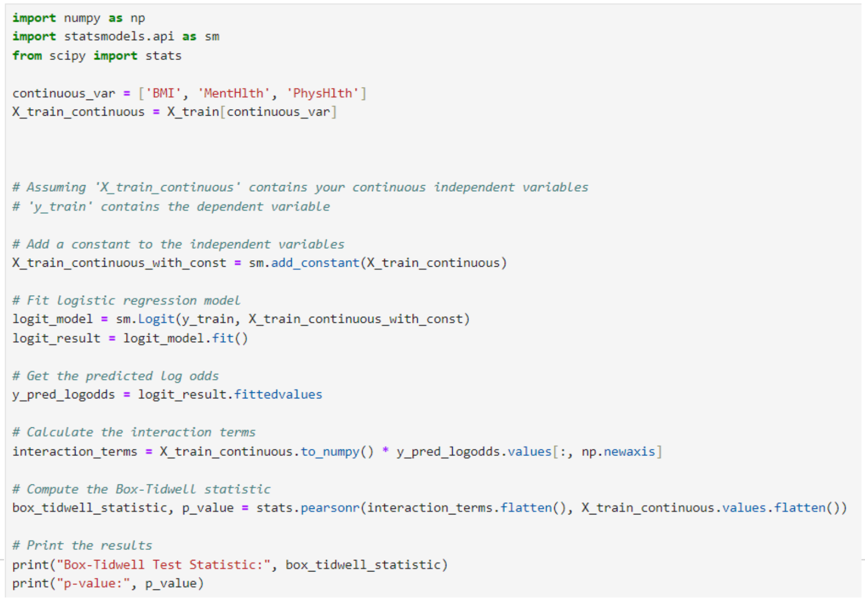


```
# make predictions on test set
y_pred=lg.predict(X_test)
y_pred1=lg.predict(X_train)
print('Training set score: {:.4f}'.format(lg.score(X_train, y_train)))

print('Test set score: {:.4f}'.format(lg.score(X_test, y_test)))
```



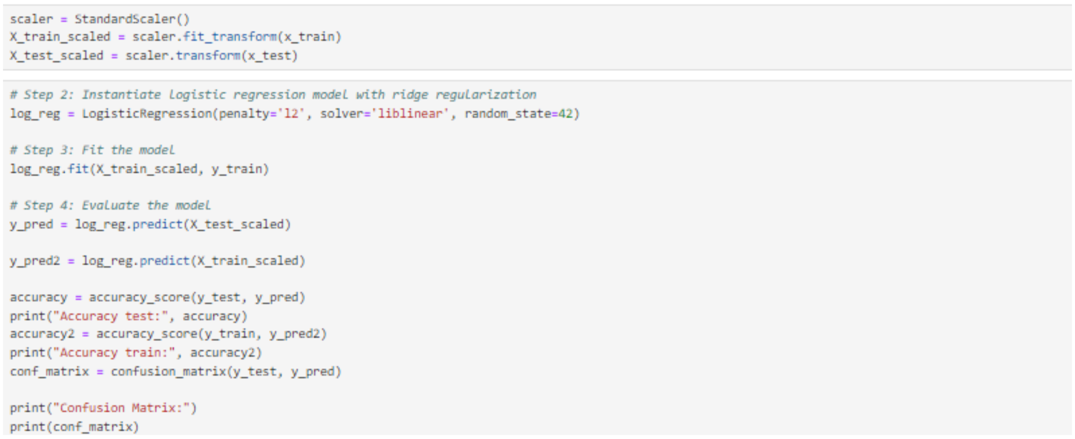
```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy test:", accuracy)
accuracy1 = accuracy_score(y_train, y_pred1)
print("Accuracy train:", accuracy1)
```

Assumption check of Logistic regression



```
import numpy as np
import statsmodels.api as sm
from scipy import stats

continuous_var = ['BMI', 'MentHlth', 'PhysHlth']
X_train_continuous = X_train[continuous_var]



# Assuming 'X_train_continuous' contains your continuous independent variables
# 'y_train' contains the dependent variable

# Add a constant to the independent variables
X_train_continuous_with_const = sm.add_constant(X_train_continuous)

# Fit logistic regression model
logit_model = sm.Logit(y_train, X_train_continuous_with_const)
logit_result = logit_model.fit()

# Get the predicted log odds
y_pred_logodds = logit_result.fittedvalues

# Calculate the interaction terms
interaction_terms = X_train_continuous.to_numpy() * y_pred_logodds.values[:, np.newaxis]

# Compute the Box-Tidwell statistic
box_tidwell_statistic, p_value = stats.pearsonr(interaction_terms.flatten(), X_train_continuous.values.flatten())

# Print the results
print("Box-Tidwell Test Statistic:", box_tidwell_statistic)
print("p-value:", p_value)
```

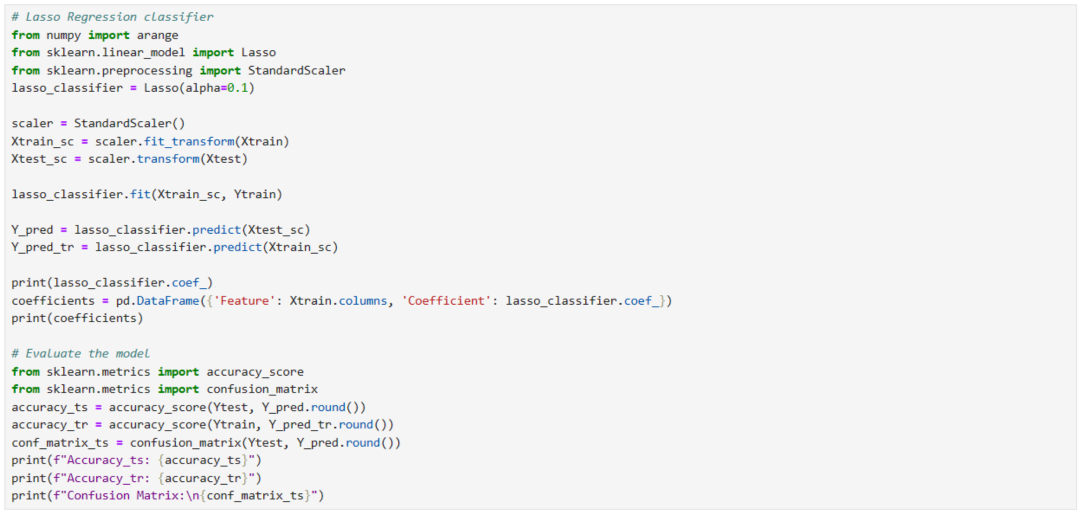
Ridge regression



```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(x_train)
X_test_scaled = scaler.transform(x_test)
```

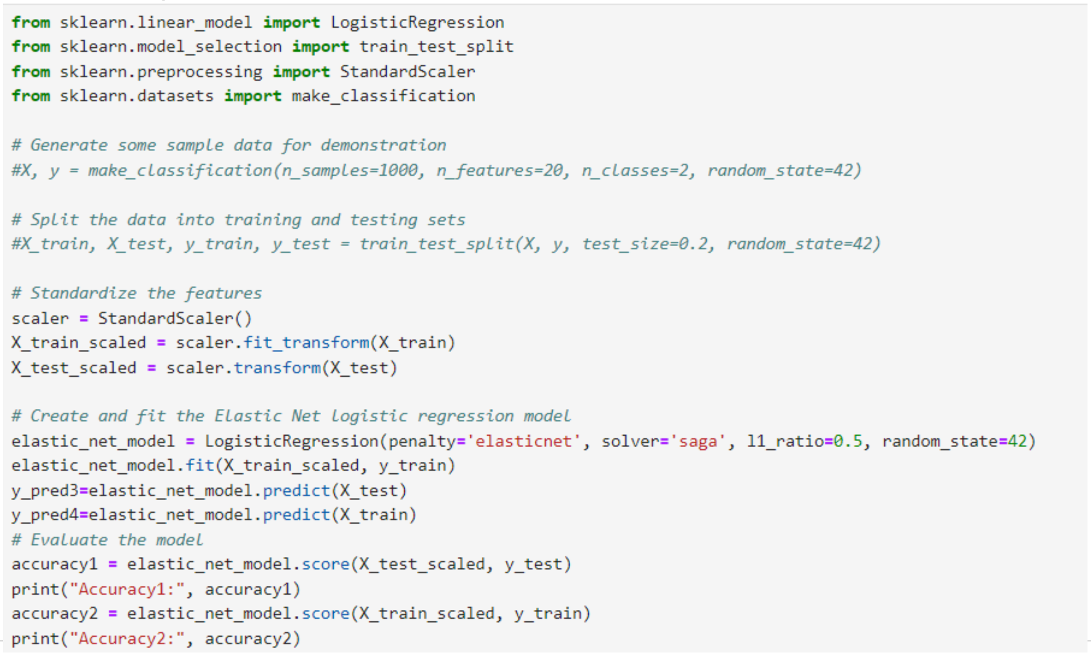
```
# Step 2: Instantiate logistic regression model with ridge regularization
log_reg = LogisticRegression(penalty='l2', solver='liblinear', random_state=42)

# Step 3: Fit the model
log_reg.fit(X_train_scaled, y_train)

# Step 4: Evaluate the model
y_pred = log_reg.predict(X_test_scaled)

y_pred2 = log_reg.predict(X_train_scaled)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy test:", accuracy)
accuracy2 = accuracy_score(y_train, y_pred2)
print("Accuracy train:", accuracy2)
conf_matrix = confusion_matrix(y_test, y_pred)

print("Confusion Matrix:")
print(conf_matrix)
```

Lasso regression



```
# Lasso Regression classifier
from numpy import arange
from sklearn.linear_model import Lasso
from sklearn.preprocessing import StandardScaler
lasso_classifier = Lasso(alpha=0.1)

scaler = StandardScaler()
Xtrain_sc = scaler.fit_transform(Xtrain)
Xtest_sc = scaler.transform(Xtest)

lasso_classifier.fit(Xtrain_sc, Ytrain)

Y_pred = lasso_classifier.predict(Xtest_sc)
Y_pred_tr = lasso_classifier.predict(Xtrain_sc)

print(lasso_classifier.coef_)
coefficients = pd.DataFrame({'Feature': Xtrain.columns, 'Coefficient': lasso_classifier.coef_})
print(coefficients)

# Evaluate the model
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
accuracy_ts = accuracy_score(Ytest, Y_pred.round())
accuracy_tr = accuracy_score(Ytrain, Y_pred_tr.round())
conf_matrix_ts = confusion_matrix(Ytest, Y_pred.round())
print(f"Accuracy_ts: {accuracy_ts}")
print(f"Accuracy_tr: {accuracy_tr}")
print(f"Confusion Matrix:\n{conf_matrix_ts}")
```

Elastic net regression



```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import make_classification

# Generate some sample data for demonstration
#X, y = make_classification(n_samples=1000, n_features=20, n_classes=2, random_state=42)

# Split the data into training and testing sets
#X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Create and fit the Elastic Net logistic regression model
elastic_net_model = LogisticRegression(penalty='elasticnet', solver='saga', l1_ratio=0.5, random_state=42)
elastic_net_model.fit(X_train_scaled, y_train)
y_pred3=elastic_net_model.predict(X_test)
y_pred4=elastic_net_model.predict(X_train)
# Evaluate the model
accuracy1 = elastic_net_model.score(X_test_scaled, y_test)
print("Accuracy1:", accuracy1)
accuracy2 = elastic_net_model.score(X_train_scaled, y_train)
print("Accuracy2:", accuracy2)
```

KNN

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train3 = sc.fit_transform(X_train)
X_test3 = sc.transform(X_test)
```

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(X_train3, y_train)
```

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

```
y_pred10 = classifier.predict(X_test3)
y_pred11 = classifier.predict(X_train3)
```

```
from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(y_test, y_pred10)
ac1 = accuracy_score(y_test,y_pred10)
ac2 = accuracy_score(y_train,y_pred11)
```

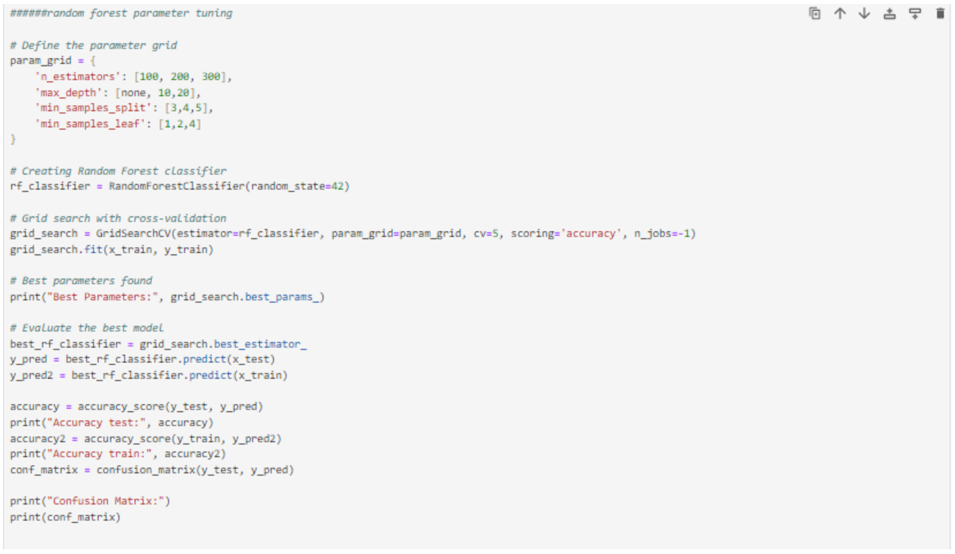
XG boost

```
# XG Boost
xgbr = xgb.XGBClassifier()
dtrain = xgb.DMatrix(Xtrain, label=Ytrain, enable_categorical=True)

# xgboost model fit with default parameters
model = xgbr.fit(Xtrain,Ytrain)
Y_pred = model.predict(Xtest)
Y_pred_tr = model.predict(Xtrain)

from sklearn.metrics import accuracy_score
tr_accuracy = accuracy_score(Ytrain,Y_pred_tr)
ts_accuracy = accuracy_score(Ytest, Y_pred)
[tr_accuracy,ts_accuracy]

# parameter tuning with grid search
from sklearn.model_selection import train_test_split, GridSearchCV
# Define hyperparameters for grid search
param_grid = {
    'n_estimators': [50, 100, 150,200,250,300,400,450,500],
    'max_depth': [3, 4, 5, 6, 7],
    'learning_rate': [0.5,0.2,0.1, 0.01, 0.001]
}
# Perform grid search with cross-validation
grid_search = GridSearchCV(estimator=xgbr, param_grid=param_grid, cv=5, scoring='accuracy')
model_xg = grid_search.fit(Xtrain, Ytrain)

# The best parameters - After tuning
best_params = model_xg.best_params_
best_params

best_xgb_model = xgb.XGBClassifier(**best_params)
best_xgb_model.fit(Xtrain,Ytrain)

Y_pred = best_xgb_model.predict(Xtest)
Y_pred_tr = best_xgb_model.predict(Xtrain)
tr_accuracy = accuracy_score(Ytrain,Y_pred_tr)
ts_accuracy = accuracy_score(Ytest, Y_pred)
[tr_accuracy,ts_accuracy]
```

Random forest

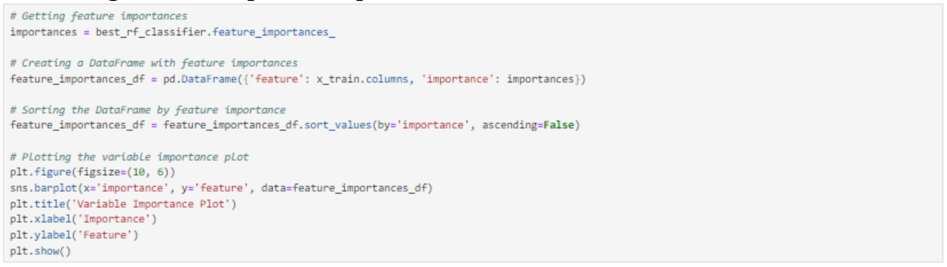
```
######random forest
from sklearn.metrics import accuracy_score
rf_classifier = RandomForestClassifier(n_estimators=100)
rf_classifier.fit(x_train,y_train)
# Evaluating the model
y_pred = rf_classifier.predict(x_test)
y_pred_tr = rf_classifier.predict(x_train)
accuracy_test = accuracy_score(y_test, y_pred)
print("Accuracy test:", accuracy_test)
accuracy_train = accuracy_score(y_train, y_pred_tr)
print("Accuracy train:", accuracy_train)

from sklearn.metrics import confusion_matrix
conf_matrix = confusion_matrix(y_test, y_pred)

print("Confusion Matrix:")
print(conf_matrix)
```

Random forest parameter tuning

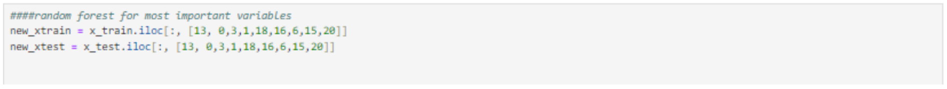


```
######random forest parameter tuning

# Define the parameter grid
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [none, 10,20],
    'min_samples_split': [3,4,5],
    'min_samples_leaf': [1,2,4]
}

# Creating Random Forest classifier
rf_classifier = RandomForestClassifier(random_state=42)

# Grid search with cross-validation
grid_search = GridSearchCV(estimator=rf_classifier, param_grid=param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(x_train, y_train)

# Best parameters found
print("Best Parameters:", grid_search.best_params_)

# Evaluate the best model
best_rf_classifier = grid_search.best_estimator_
y_pred = best_rf_classifier.predict(x_test)
y_pred2 = best_rf_classifier.predict(x_train)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy test:", accuracy)
accuracy2 = accuracy_score(y_train, y_pred2)
print("Accuracy train:", accuracy2)
conf_matrix = confusion_matrix(y_test, y_pred)

print("Confusion Matrix:")
print(conf_matrix)
```

Obtaining feature importance plot

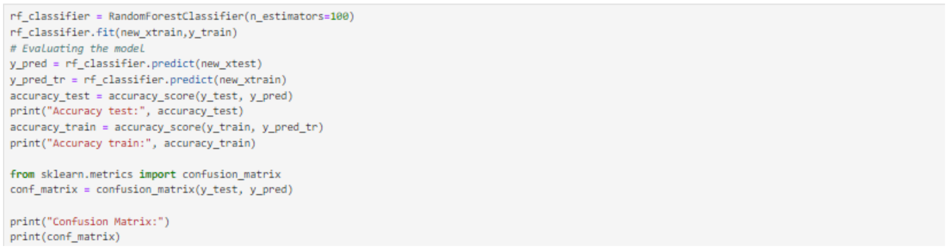


```
# Getting feature importances
importances = best_rf_classifier.feature_importances_

# Creating a DataFrame with feature importances
feature_importances_df = pd.DataFrame({'feature': x_train.columns, 'importance': importances})

# Sorting the DataFrame by feature importance
feature_importances_df = feature_importances_df.sort_values(by='importance', ascending=False)

# Plotting the variable importance plot
plt.figure(figsize=(10, 6))
sns.barplot(x='importance', y='feature', data=feature_importances_df)
plt.title('Variable Importance Plot')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.show()
```



```
####random forest for most important variables
new_xtrain = x_train.iloc[:, [13, 0,3,1,18,16,6,15,20]]
new_xtest = x_test.iloc[:, [13, 0,3,1,18,16,6,15,20]]
```

Fit random forest for most important features



```
rf_classifier = RandomForestClassifier(n_estimators=100)
rf_classifier.fit(new_xtrain,y_train)
# Evaluating the model
y_pred = rf_classifier.predict(new_xtest)
y_pred_tr = rf_classifier.predict(new_xtrain)
accuracy_test = accuracy_score(y_test, y_pred)
print("Accuracy test:", accuracy_test)
accuracy_train = accuracy_score(y_train, y_pred_tr)
print("Accuracy train:", accuracy_train)

from sklearn.metrics import confusion_matrix
conf_matrix = confusion_matrix(y_test, y_pred)

print("Confusion Matrix:")
print(conf_matrix)
```

Parameter tuning

```
###optimize new random forest model
######random forest parameter tuning

# Define the parameter grid
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20],
    'min_samples_split': [3,4,5],
    'min_samples_leaf': [1, 2, 4]
}

# Creating Random Forest classifier
rf_classifier = RandomForestClassifier(random_state=42)

# Grid search with cross-validation
grid_search = GridSearchCV(estimator=rf_classifier, param_grid=param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(new_xtrain, y_train)

# Best parameters found
print("Best Parameters:", grid_search.best_params_)

# Evaluate the best model
best_rf_classifier = grid_search.best_estimator_
y_pred = best_rf_classifier.predict(new_xtest)
y_pred2 = best_rf_classifier.predict(new_xtrain)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy test:", accuracy)
accuracy2 = accuracy_score(y_train, y_pred2)
print("Accuracy train:", accuracy2)
conf_matrix = confusion_matrix(y_test, y_pred)

print("Confusion Matrix:")
print(conf_matrix)
```

Partial dependency plot

```
###optimize new random forest model
######random forest parameter tuning

# Define the parameter grid
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20],
    'min_samples_split': [3,4,5],
    'min_samples_leaf': [1, 2, 4]
}

# Creating Random Forest classifier
rf_classifier = RandomForestClassifier(random_state=42)

# Grid search with cross-validation
grid_search = GridSearchCV(estimator=rf_classifier, param_grid=param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(new_xtrain, y_train)

# Best parameters found
print("Best Parameters:", grid_search.best_params_)

# Evaluate the best model
best_rf_classifier = grid_search.best_estimator_
y_pred = best_rf_classifier.predict(new_xtest)
y_pred2 = best_rf_classifier.predict(new_xtrain)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy test:", accuracy)
accuracy2 = accuracy_score(y_train, y_pred2)
print("Accuracy train:", accuracy2)
conf_matrix = confusion_matrix(y_test, y_pred)

print("Confusion Matrix:")
print(conf_matrix)
```

```
from sklearn import inspection
```

```
from sklearn.inspection import PartialDependenceDisplay
```

```
features_to_plot =[(0,)]   # Example: Plotting partial dependence for the feature at index 3

# Plot partial dependence
fig, ax = plt.subplots(figsize=(15, 10))
PartialDependenceDisplay.from_estimator(best_rf_classifier, new_xtest, features_to_plot, ax=ax)
plt.title("Partial Dependence Plots")
plt.show()
```

```
features_to_plot =[(1,)]   # Example: Plotting partial dependence for the feature at index 3
```

```
features_to_plot =[(0,)]   # Example: Plotting partial dependence for the feature at index 3

# Plot partial dependence
fig, ax = plt.subplots(figsize=(15, 10))
PartialDependenceDisplay.from_estimator(best_rf_classifier, new_xtest, features_to_plot, ax=ax)
plt.title("Partial Dependence Plots")
plt.show()
```

```
features_to_plot =[(1,)]   # Example: Plotting partial dependence for the feature at index 3

# Plot partial dependence
fig, ax = plt.subplots(figsize=(15, 10))
PartialDependenceDisplay.from_estimator(best_rf_classifier, new_xtest, features_to_plot, ax=ax)
plt.title("Partial Dependence Plots")
plt.show()
```

```
features_to_plot =[(2,)]   # Example: Plotting partial dependence for the feature at index 3

# Plot partial dependence
fig, ax = plt.subplots(figsize=(15, 10))
PartialDependenceDisplay.from_estimator(best_rf_classifier, new_xtest, features_to_plot, ax=ax)
plt.title("Partial Dependence Plots")
plt.show()
```

```
features_to_plot =[(8,)]   # Example: Plotting partial dependence for the feature at index 3

# Plot partial dependence
fig, ax = plt.subplots(figsize=(15, 10))
PartialDependenceDisplay.from_estimator(best_rf_classifier, new_xtest, features_to_plot, ax=ax)
plt.title("Partial Dependence Plots")
plt.show()
```