Data Analysis and Prediction with Python

Diabetes Analysis and Prediction

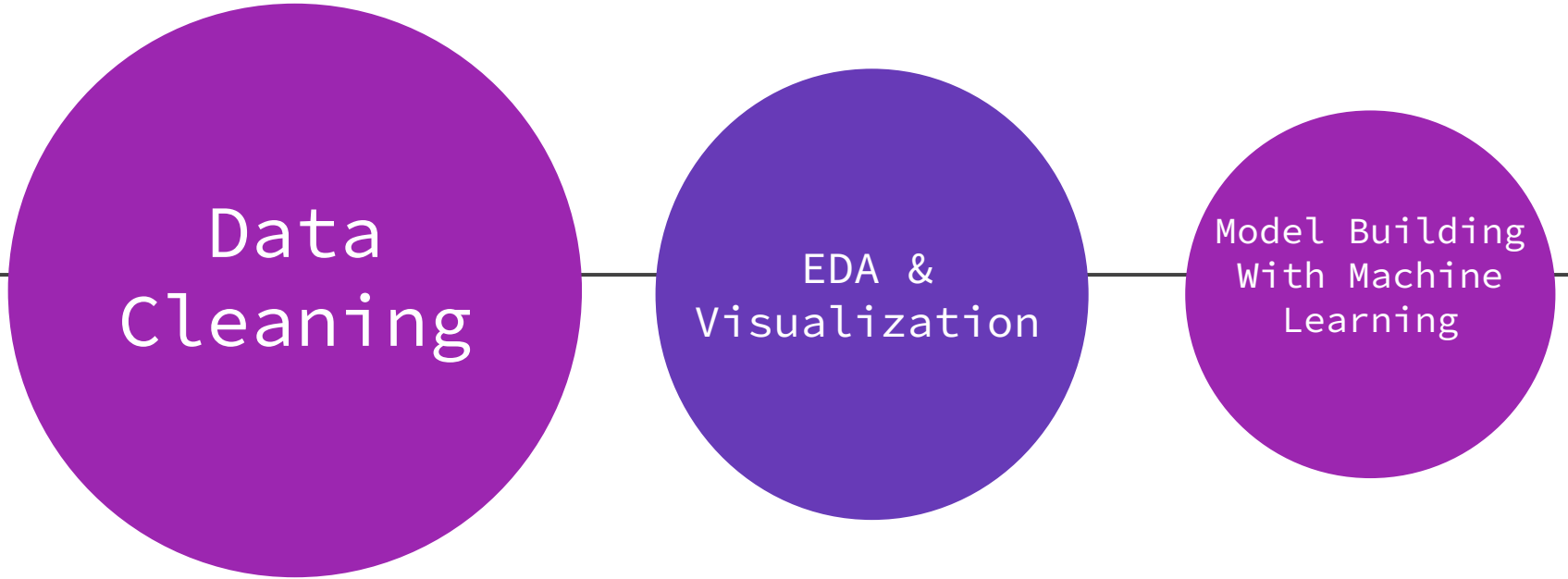
About the project

In this project, there are three parts.

Part 1: Data cleaning and transformation.

Part 2: Exploratory Data Analysis with Visualizations

Part 3: Model Building for Diabetes Prediction


Data Cleaning
EDA & Visualization
Model Building With Machine Learning

Part 1: Data cleaning and transformation.

The main goal is to clean the data using advanced python algorithm. The original data had missing values, the columns that had inconsistent entries and some special character that needed to be manipulated.

In addition, necessary data transformation (merging the tables) is completed for the purpose of this project.

Sample codes for data cleaning

```
#creating one-hot-encoding dummy variables for the cholesterol level labels.
chol_value_list = diabetes_train['gluc'].unique()

diabetes_train['chol_low'] = ( diabetes_train['cholesterol'] == 'low' ).astype(float)
diabetes_train['chol_high'] = ( diabetes_train['cholesterol'] == 'high' ).astype(float)
diabetes_train['chol_medium'] = ( diabetes_train['cholesterol'] == 'medium' ).astype(float)

diabetes_test['chol_low'] = ( diabetes_test['cholesterol'] == 'low' ).astype(float)
diabetes_test['chol_high'] = ( diabetes_test['cholesterol'] == 'high' ).astype(float)
diabetes_test['chol_medium'] = ( diabetes_test['cholesterol'] == 'medium' ).astype(float)

diabetes_train.head()
```

Sample codes for data cleaning (Continuation)

```
#creating a function that creates one-hot-encoding dummy variables for the purpose of reusing
def one_hot_encoder( df, col_name, cat_values ):
    for i in list(cat_values):
        df[f"{col_name}_{i}"] = ( df[col_name] == i ).astype(float)
    return df
```

```
#using one_hot_encoder function for the glucose level label
diabetes_train=one_hot_encoder(diabetes_train, 'gluc', ['low','high', 'medium'] )
diabetes_test=one_hot_encoder(diabetes_test, 'gluc', ['low','high', 'medium'] )
diabetes_train.head()
```

Sample codes for data cleaning (Continuation)

```
#separating blood pressure using the lambda function and created two columns for high and low blood pressure measures
diabetes_train['press_high'] = diabetes_train['pressure'].apply( lambda x: x.split("/")[0] ).astype(float)
diabetes_test['press_high'] = diabetes_test['pressure'].apply( lambda x: x.split("/")[0] ).astype(float)

diabetes_train['press_low'] = diabetes_train['pressure'].apply( lambda x: x.split("/")[1] ).astype(float)
diabetes_test['press_low'] = diabetes_test['pressure'].apply( lambda x: x.split("/")[1] ).astype(float)
```

```
#cleaning the gender column the training data

diabetes_info_train.loc[ diabetes_info_train['gender']=='female', 'gender' ] = 'f'
diabetes_info_train.loc[ diabetes_info_train['gender']=='male', 'gender' ] = 'm'
gender_uniq_vals = diabetes_info_train['gender'].unique()
diabetes_info_train = one_hot_encoder( diabetes_info_train, "gender", gender_uniq_vals )
```

Sample codes for data cleaning (Continuation)

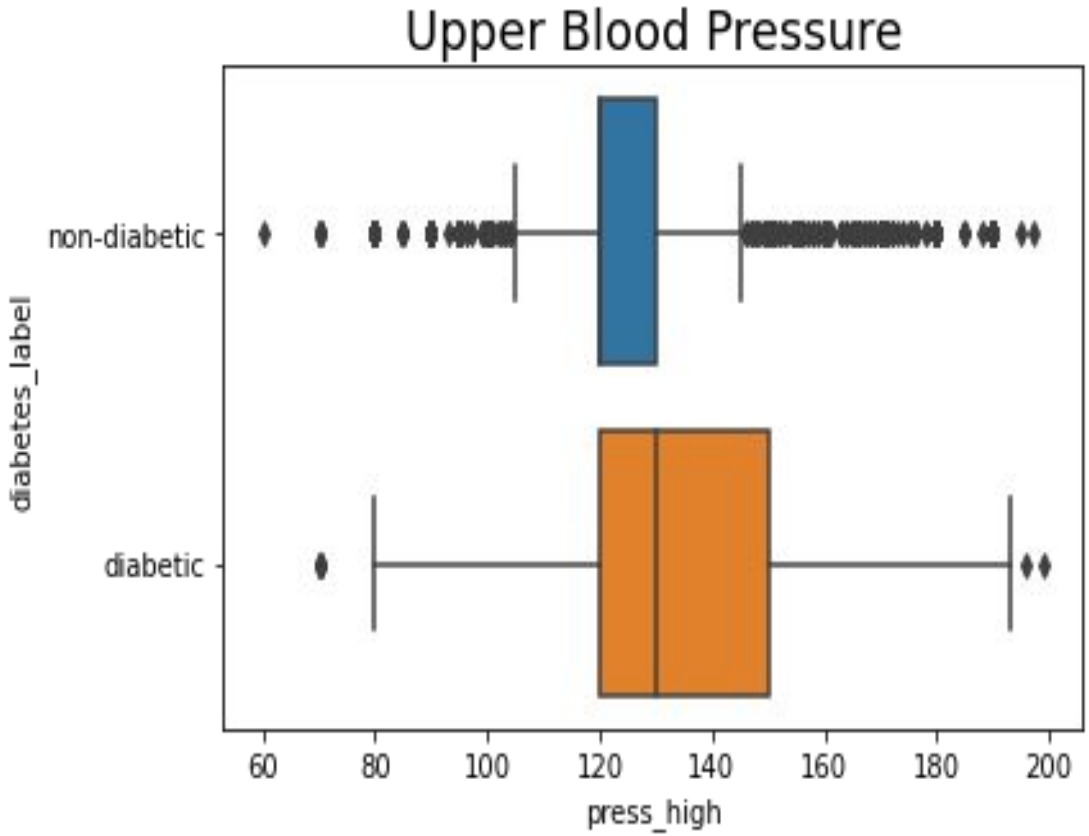
```
#cleaning the age column for the test data
diabetes_info_test['age_2'] = diabetes_info_test['age']
diabetes_info_test.loc[ diabetes_info_test['age'] > 150, 'age_2'] = diabetes_info_test['age'] / 365
diabetes_info_test['age'] = diabetes_info_test['age'].apply( lambda x: math.floor( x ) )
diabetes_info_test.head()
```

```
#the weight column has too many missing values, so replaced the missing weigths with average weigth.
imp = SimpleImputer(missing_values=np.nan, strategy='mean')

imp.fit( diabetes_info_train[['weight']] )

diabetes_info_train['weight_2']  = imp.transform( diabetes_info_train[['weight']]  )
diabetes_info_test['weight_2']  = imp.transform( diabetes_info_test[['weight']]  )
```

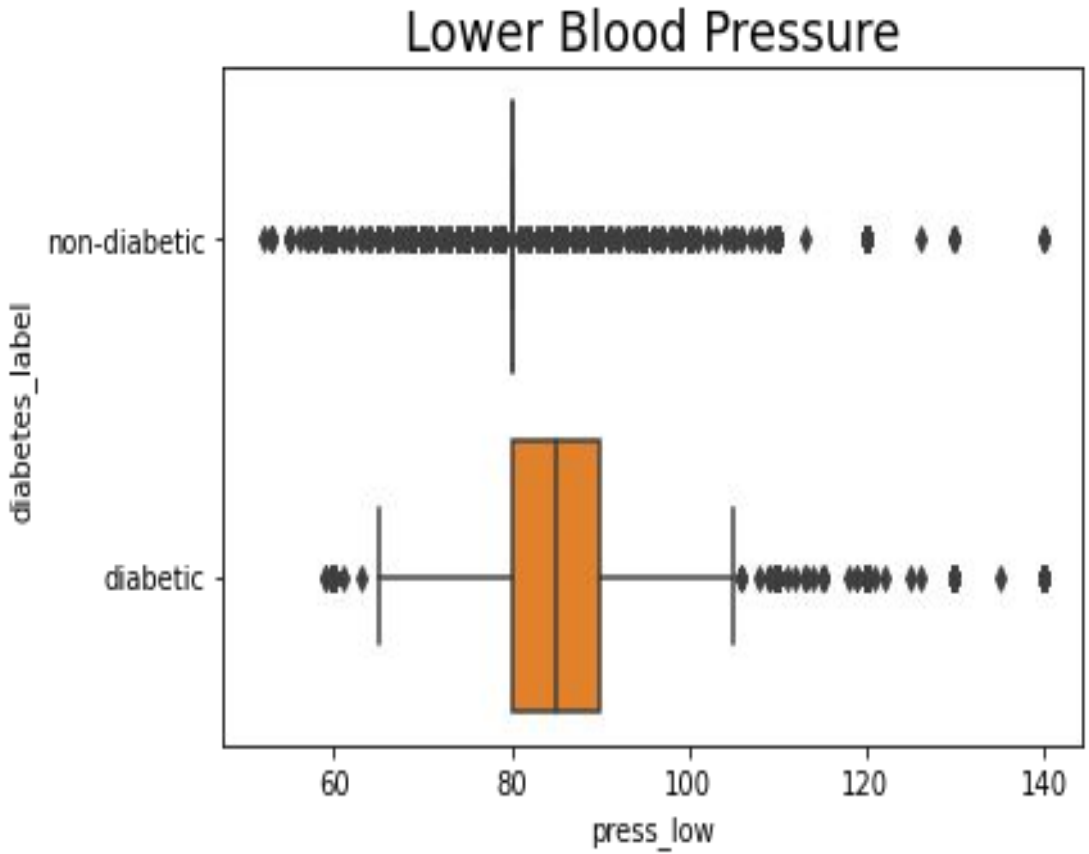
Part 2: Exploratory Data Analysis with Visualizations

In this part, I conducted Exploratory Data Analysis (EDA) for six variables and then created the visualizations (pie chart, bar/stacked bar charts, histogram, boxplot) for each variable using matplotlib.

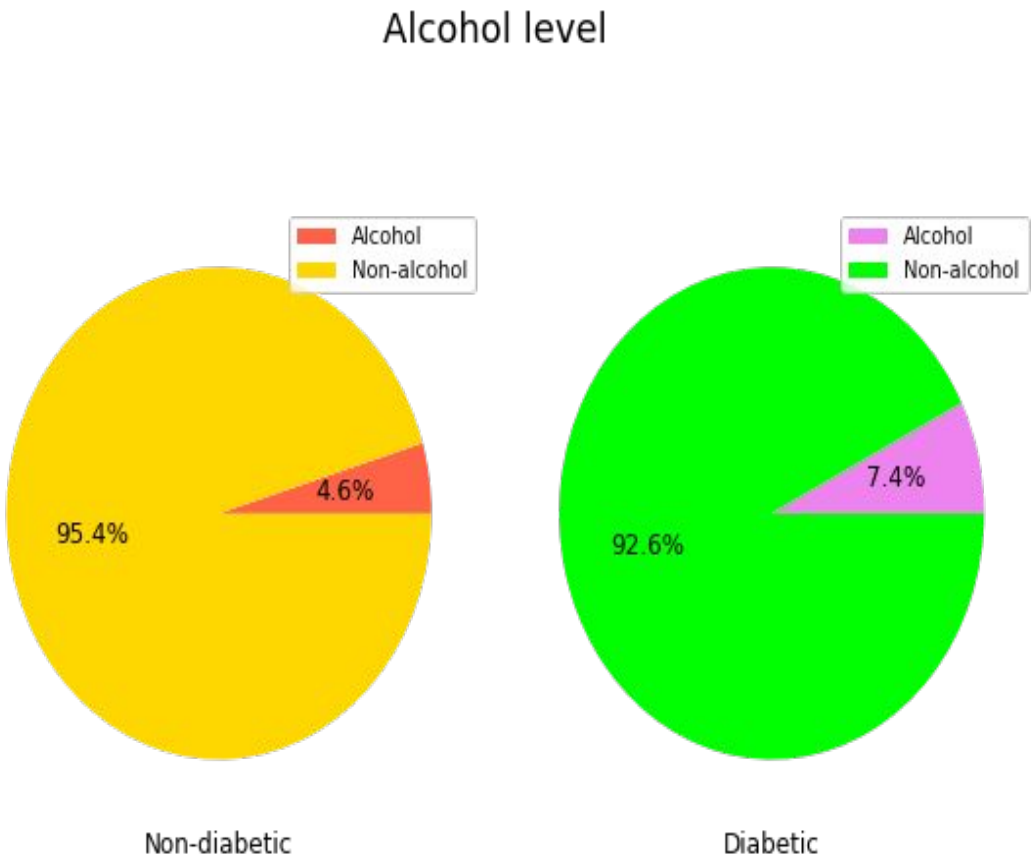
Based on the EDA,in terms of blood glucose level, activity level, weight and blood pressure, significant differences have shown between diabetic population and non-diabetic population. However, in terms of alcohol and smoking, there was not significant difference.



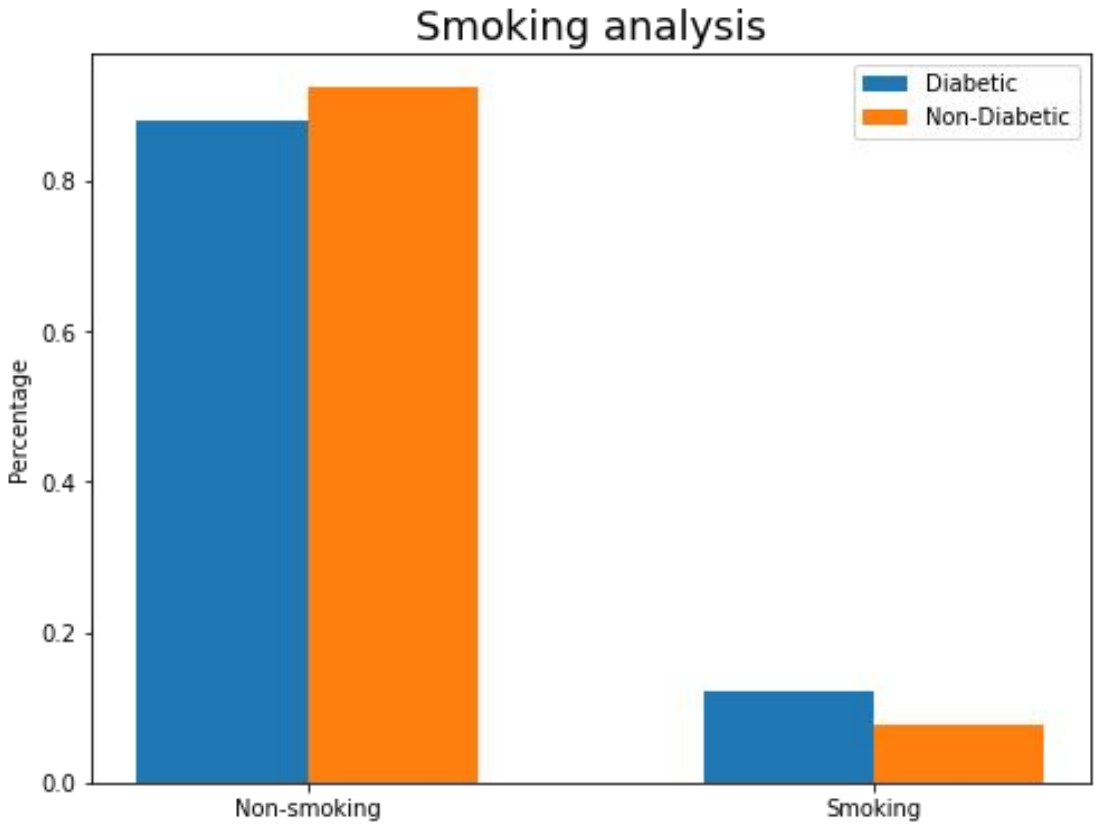
- Diabetic people tend to have higher upper blood pressures
- Diabetic people have a wider range of values regarding the upper blood pressure



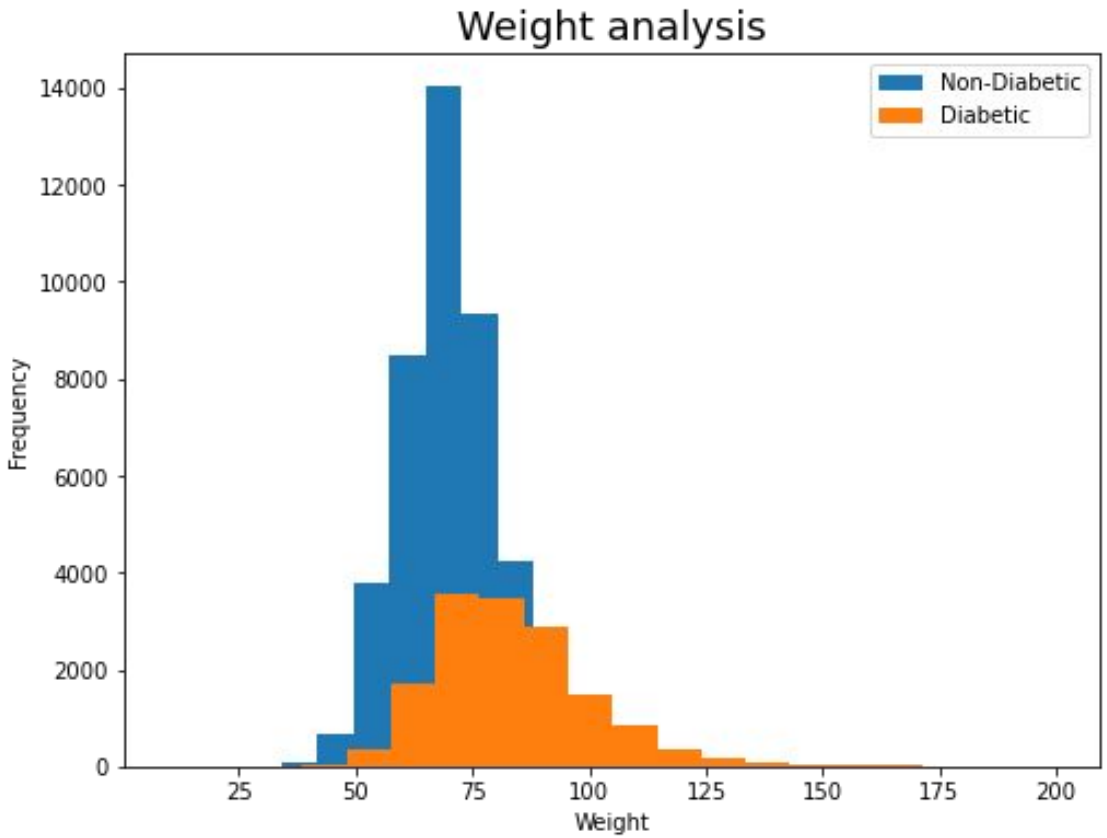
- Diabetic people tend to have higher lower blood pressures
- Diabetic people have a wider range of values regarding the lower blood pressure



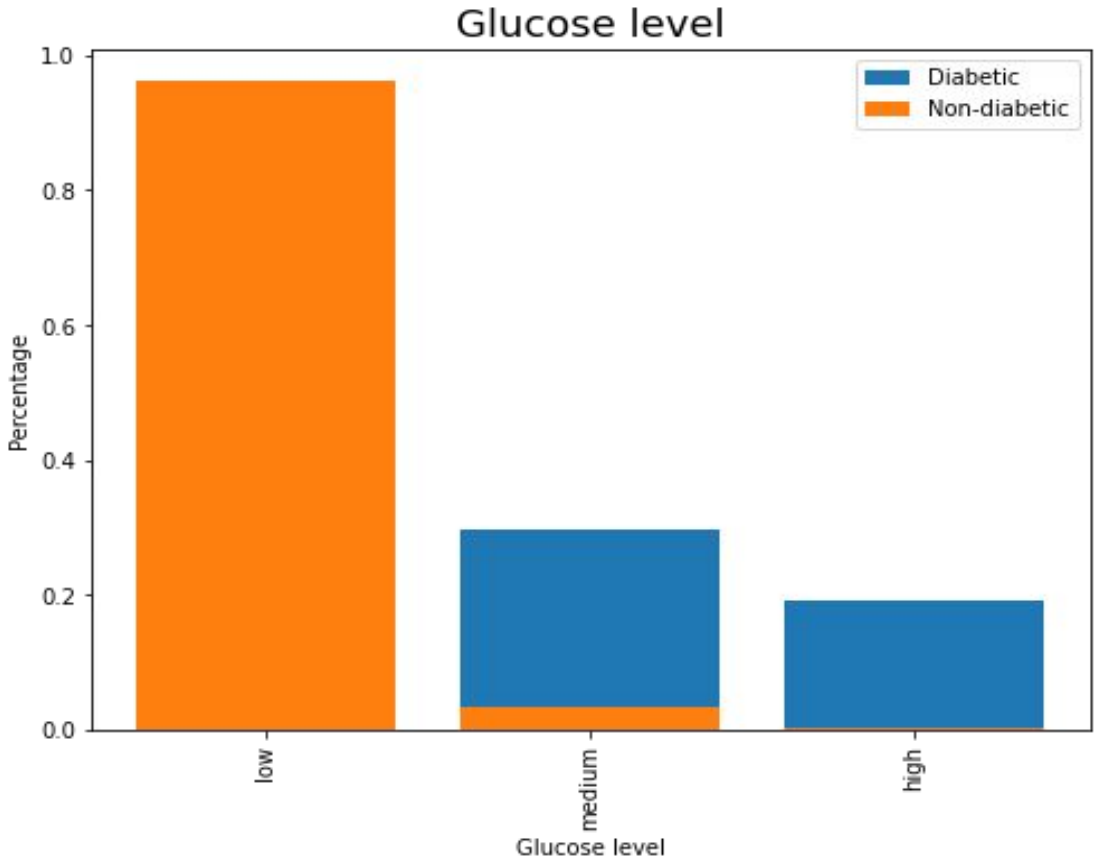
- For alcohol, diabetic people have slightly higher rates for alcohol consumption compared with non-diabetic people.



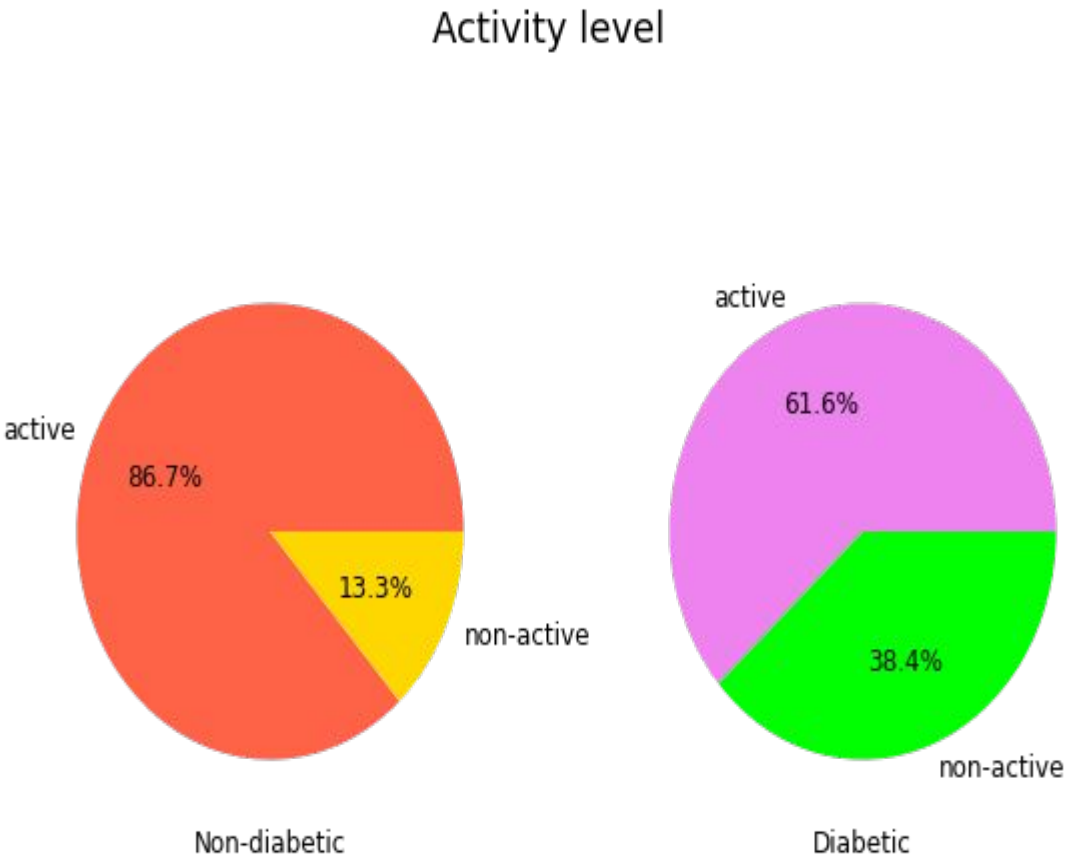
- For smoking, diabetic people have slightly higher rates for smoking consumption compared with non-diabetic people.



- Visualization for weight distribution across diabetic and non-diabetic people indicates that diabetic people tend to have higher weights relative to non-diabetic people.



- Visualization for glucose level distribution demonstrates that diabetic people have higher glucose level in average than non-diabetic people.



- Physical activity visualization showed that non-diabetic people are more physically active than diabetic people in average.

Part 3: Model Building for Diabetes Prediction

In this part, I conducted two different machine learning models and predicted the test data results with high level of accuracy (98.88%).

In addition, displayed feature importance visualization which provided clear insight of the each feature.

Logistic regression model:

I built the logistic regression model as the first machine learning model. Target variable is the dummy variable indicating diabetic or non-diabetic. Regressors or explanatory variables are smoking, alcohol consumption, upper blood pressure, lower blood pressure, cholesterol level, glucose level, height, gender, age, and weight.

Accuracy Score: 96.41%

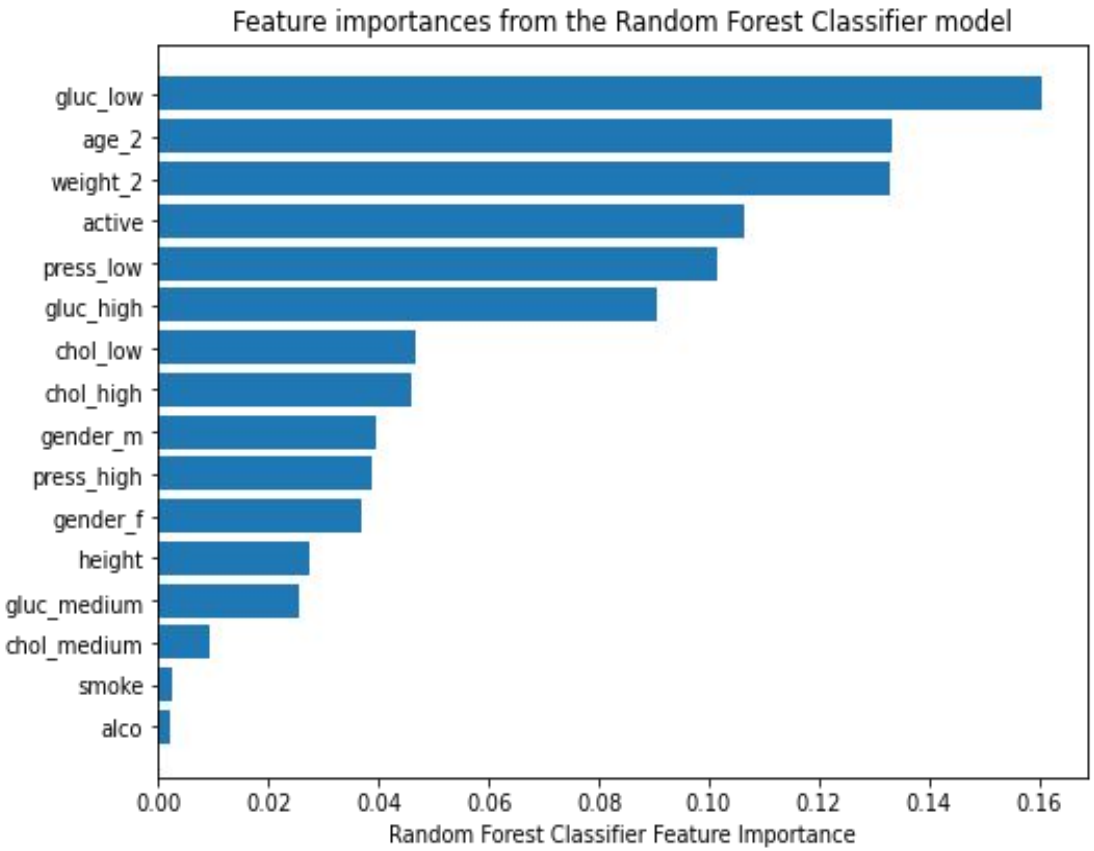
```
#using the Logistic Regression
clf_logistic = LogisticRegression(random_state=10).fit( diabetes_train_x, diabetes_train_y )

diabetes_test_pred = clf_logistic.predict( diabetes_test_x )

from sklearn.metrics import accuracy_score

accuracy_score( diabetes_test_pred, diabetes_test_y )
```

Random Forest Classifier model:

Target variable is the dummy variable indicating diabetic or non-diabetic. Regressors or explanatory variables are smoking, alcohol consumption, upper blood pressure, lower blood pressure, cholesterol level, glucose level, height, gender, age, and weight. Max depth of each tree in the forest is 19 and random state seed value is 0.

Accuracy Score: 98.88%

```
#using Random Forest
clf_RF = RandomForestClassifier(max_depth=19, random_state=0).fit( diabetes_train_x, diabetes_train_y )

diabetes_test_pred_rf = clf_RF.predict( diabetes_test_x )

accuracy_score( diabetes_test_pred_rf, diabetes_test_y )
```



- Plotted the feature importances for explanatory variables from this Random Forest Classifier model.
- The model indicated that glucose level, age, weight, physical activity, lower blood pressure, upper blood pressure, and gender are important variables in diagnosing the diabetes.

Conclusion

- Based on the EDA,in terms of blood glucose level, activity level, weight and blood pressure, significant differences have shown between diabetic population and non-diabetic population. However, in terms of alcohol and smoking, there was not significant difference.
- Based on machine learning prediction models, glucose level, age, weight, physical activity, lower blood pressure, upper blood pressure, and gender are important variables in diagnosing the diabetes.
- Random Forest Classifier model reached 98.88% accuracy after data cleaning and transformation process.