335_final (4)

December 15, 2024

#Edwin Arroyo #1. The application domains involved in this project are the varous algorithms that we have learned throughout the course. In particular I will be using Logisic regression. The goal with this model is to try and build a predictive model to see if you can assess a person's lifestyle habits and their health markers to see if they are at risk of getting diabtes.

[47]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix, roc_auc_score, roc_curve
from sklearn.tree import DecisionTreeClassifier
import graphviz
from sklearn import tree
from sklearn.tree import plot_tree
from sklearn.svm import LinearSVC
from sklearn.linear_model import LinearRegression
import seaborn as sns
```

#2. The attributes available in the data set: Diabetes_012, HighBP, HighCol, ColCheck, BMI, Smoker, Stroke, HeartDisearOrAttack, PhysActivity, Fruits, Veggies, HvyAlcoholConsump, Any-Healthcare, NoDocbcCose, GenHlth, MenHlth, PhyHlth, DiffWalk, Sex. Age. Education, Income

The subset of attributes that I will be using in order to build this predictive model are HighBP, HighCol, BMI, PhysActivity, DiffWalk. The reason that I chose these attributes is beacuse I think they would be the most indicitive to whether or not a person is at risk of getting diabetes, if these markers are not in the normal range.

Most of the attributes that I will be using are in a binary format e.g HighCol: 0 means that the person does not have high cholesterol and 1 means that the person does have high cholsetorol. The exception to this is BMI which is measured using float numbers. To have the least amount of complications as possible if the BMI is x.5 or over it will be rounded to the nearest whole number. The way in whcih I wll turn this attrute into binary format will be by turnign any BMI < 25 into 0 meaning that the person is not overwight, and turning any BMI > 25 into 1, indicating that the person is overwight.

The target attribute will be Diabetes_012 in which 0 indicates that the preson does not have

diabetes, 1 indicates that the person is pre-diabetic, and 2 indicates that the person has diabetes. I will split this up into 0 and 1, where 0 indicates that the person does not have diabetes and 1 indicates that the person is either pre-diabetic or the person has diabetes.

From what I can see, this dataset does not contain any missing or null values, but I will still conduct the checks and if any of the attributes do conatian null or missing values I will most likely fill it with the mean of that attribute.

#3. Based on analyziing the data and drawing from my basic knowledge of health, I assume that the people whose healthmarkers (feature attributes) are higher than normal are at a greater risk of getting diabetes. normal here meaning average, but in this case the markers are in a binary format so 0 would indicate the person's healthmarkers are in the normal range, whereas a 1 would indicate the person's healthmarkers are higher than that of the normal range.

#4. Cleanse and preprocess data

[48]:
```
# read in the file that will be used for training and testing
data = pd.read_csv('data.csv')

# clean and pre-process the data

# remove columns that I will not be using for readability
data = data.drop(columns=["Smoker"])
data = data.drop(columns=["Education"])
data = data.drop(columns=["HeartDiseaseorAttack"])
data = data.drop(columns=["AnyHealthcare"])
data = data.drop(columns=["MentHlth"])
data = data.drop(columns=["Fruits"])
data = data.drop(columns=["NoDocbcCost"])
data = data.drop(columns=["Sex"])
data = data.drop(columns=["HvyAlcoholConsump"])
data = data.drop(columns=["Veggies"])
data = data.drop(columns=["Income"])
data = data.drop(columns=["PhysHlth"])


# fill in missing data with the mean just in case I did not catch a missing␣
 ↪value
data["Diabetes_binary"] = data["Diabetes_binary"].
 ↪fillna(data["Diabetes_binary"].mean())
data["HighBP"] = data["HighBP"].fillna(data["HighBP"].mean())
data["HighChol"] = data["HighChol"].fillna(data["HighChol"].mean())
data["CholCheck"] = data["CholCheck"].fillna(data["CholCheck"].mean())
data["BMI"] = data["BMI"].fillna(data["BMI"].mean())
data["Stroke"] = data["Stroke"].fillna(data["Stroke"].mean())
data["PhysActivity"] = data["PhysActivity"].fillna(data["PhysActivity"].mean())
data["GenHlth"] = data["GenHlth"].fillna(data["GenHlth"].mean())
#data["PhysHlth"] = data["PhysHlth"].fillna(data["PhysHlth"].mean())
data["DiffWalk"] = data["DiffWalk"].fillna(data["DiffWalk"].mean())
```

```
# map data that needs to be in binary format

# if a persons BMI is under 25 then value will be 0 : If persons BMI is over 25␣
 ↪value will be 1
data["BMI"] = data["BMI"].apply(lambda x: 0 if x < 25 else 1)

# split diabtes_012 into 0 and 1 : 0 = person does not have diabets : 1 =␣
 ↪person is either pre-diabetic or has diabetes
data['Diabetes_risk'] = data['Diabetes_binary'].map({0: 0, 1: 1, 2: 1})


# split GenHlth : 1-3 = helathy : 4-5 = unhealthy
data['GenHlth'] = data['GenHlth'].map({1: 0, 2: 0, 3: 0, 4: 1, 5: 1})

# AGE - 18-44 = 0, 45 and above = 1
data['Age'] = data['Age'].apply(lambda x: 0 if x <= 5 else 1)

# summary statistics
summary_stats = data.describe()
print("Summary Statistics: \n")
print(summary_stats)

# Compute the covariance matrix
cov_matrix = data.cov()
print("\nCovariance Matrix: \n")
print(cov_matrix)

# compute correlation matrix
correlation_matrix = data.corr()
print("\nCorrelation Matrix: \n")
print(correlation_matrix)

# visualize correlation matrix
plt.figure(figsize=(10, 8))  # Adjust the size of the plot
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f',␣
 ↪linewidths=0.5)

# Display the plot
plt.title('Correlation Matrix')
plt.show()


risk_factors = ['High BMI', 'High Blood Pressure', 'High Cholesterol', 'Poor␣
 ↪General Health']
prevalence = [
    data['BMI'].mean() * 100,
```

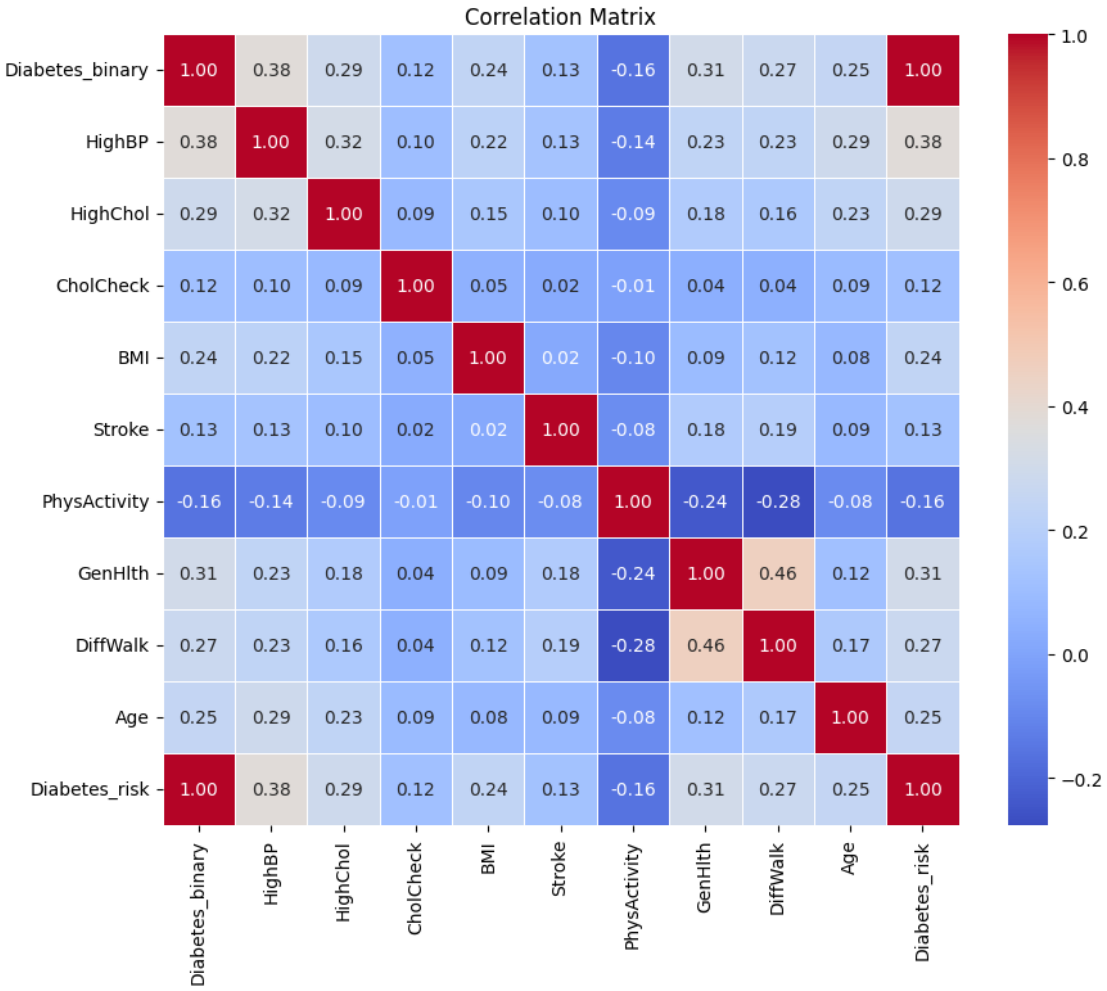
```
    data['HighBP'].mean() * 100,
    data['HighChol'].mean() * 100,
    data['GenHlth'].mean() * 100
]

# Create the bar plot
plt.figure(figsize=(8, 6))
plt.bar(risk_factors, prevalence, color='skyblue')
plt.title('Prevalence of Diabetes Risk Factors')
plt.xlabel('Risk Factor')
plt.ylabel('Prevalence (%)')
plt.ylim(0, 100)
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
plt.close()
```

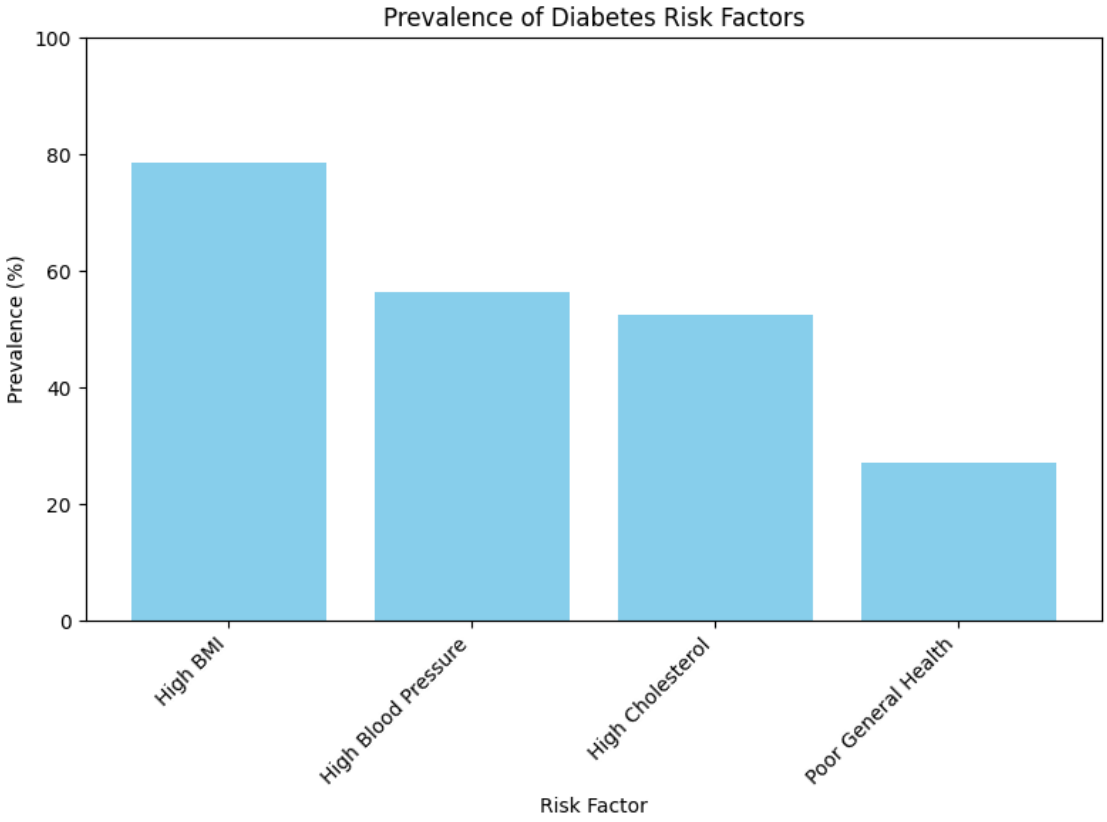
Summary Statistics:

```
       Diabetes_binary        HighBP      HighChol     CholCheck  \
count     70692.000000  70692.000000  70692.000000  70692.000000
mean          0.500000      0.563458      0.525703      0.975259
std           0.500004      0.495960      0.499342      0.155336
min           0.000000      0.000000      0.000000      0.000000
25%           0.000000      0.000000      0.000000      1.000000
50%           0.500000      1.000000      1.000000      1.000000
75%           1.000000      1.000000      1.000000      1.000000
max           1.000000      1.000000      1.000000      1.000000

                BMI        Stroke  PhysActivity       GenHlth      DiffWalk  \
count  70692.000000  70692.000000  70692.000000  70692.000000  70692.000000
mean       0.786213      0.062171      0.703036      0.270342      0.252730
std        0.409981      0.241468      0.456924      0.444139      0.434581
min        0.000000      0.000000      0.000000      0.000000      0.000000
25%        1.000000      0.000000      0.000000      0.000000      0.000000
50%        1.000000      0.000000      1.000000      0.000000      0.000000
75%        1.000000      0.000000      1.000000      1.000000      1.000000
max        1.000000      1.000000      1.000000      1.000000      1.000000

                Age  Diabetes_risk
count  70692.000000   70692.000000
mean       0.848116       0.500000
std        0.358911       0.500004
min        0.000000       0.000000
25%        1.000000       0.000000
50%        1.000000       0.500000
75%        1.000000       1.000000
```

```
max        1.000000       1.000000

Covariance Matrix:

                 Diabetes_binary    HighBP  HighChol  CholCheck       BMI  \
Diabetes_binary         0.250004  0.094609  0.072209   0.008962  0.048938
HighBP                  0.094609  0.245977  0.078386   0.007957  0.044398
HighChol                0.072209  0.078386  0.249343   0.006669  0.030442
CholCheck               0.008962  0.007957  0.006669   0.024129  0.003170
BMI                     0.048938  0.044398  0.030442   0.003170  0.168084
Stroke                  0.015143  0.015456  0.012032   0.000845  0.002060
PhysActivity           -0.036249 -0.030843 -0.020638  -0.000586 -0.019154
GenHlth                 0.068078  0.051432  0.038906   0.002898  0.017140
DiffWalk                0.059244  0.050604  0.035164   0.002999  0.022202
Age                     0.044695  0.051051  0.042077   0.004942  0.011749
Diabetes_risk           0.250004  0.094609  0.072209   0.008962  0.048938

                   Stroke  PhysActivity   GenHlth  DiffWalk       Age  \
Diabetes_binary  0.015143     -0.036249  0.068078  0.059244  0.044695
HighBP           0.015456     -0.030843  0.051432  0.050604  0.051051
HighChol         0.012032     -0.020638  0.038906  0.035164  0.042077
CholCheck        0.000845     -0.000586  0.002898  0.002999  0.004942
BMI              0.002060     -0.019154  0.017140  0.022202  0.011749
Stroke           0.058307     -0.008825  0.018939  0.020176  0.007448
PhysActivity    -0.008825      0.208779 -0.048305 -0.054978 -0.013602
GenHlth          0.018939     -0.048305  0.197260  0.088471  0.018343
DiffWalk         0.020176     -0.054978  0.088471  0.188860  0.025952
Age              0.007448     -0.013602  0.018343  0.025952  0.128817
Diabetes_risk    0.015143     -0.036249  0.068078  0.059244  0.044695

                 Diabetes_risk
Diabetes_binary       0.250004
HighBP                0.094609
HighChol              0.072209
CholCheck             0.008962
BMI                   0.048938
Stroke                0.015143
PhysActivity         -0.036249
GenHlth               0.068078
DiffWalk              0.059244
Age                   0.044695
Diabetes_risk         0.250004

Correlation Matrix:

                 Diabetes_binary    HighBP  HighChol  CholCheck       BMI  \
Diabetes_binary         1.000000  0.381516  0.289213   0.115382  0.238733
HighBP                  0.381516  1.000000  0.316515   0.103283  0.218350
```

```
HighChol               0.289213  0.316515  1.000000   0.085981  0.148698
CholCheck              0.115382  0.103283  0.085981   1.000000  0.049776
BMI                    0.238733  0.218350  0.148698   0.049776  1.000000
Stroke                 0.125427  0.129060  0.099786   0.022529  0.020804
PhysActivity          -0.158666 -0.136102 -0.090453  -0.008249 -0.102248
GenHlth                0.306559  0.233487  0.175427   0.041998  0.094132
DiffWalk               0.272646  0.234784  0.162043   0.044430  0.124614
Age                    0.249054  0.286795  0.234780   0.088643  0.079848
Diabetes_risk          1.000000  0.381516  0.289213   0.115382  0.238733

                     Stroke  PhysActivity   GenHlth  DiffWalk       Age  \
Diabetes_binary  0.125427     -0.158666  0.306559  0.272646  0.249054
HighBP           0.129060     -0.136102  0.233487  0.234784  0.286795
HighChol         0.099786     -0.090453  0.175427  0.162043  0.234780
CholCheck        0.022529     -0.008249  0.041998  0.044430  0.088643
BMI              0.020804     -0.102248  0.094132  0.124614  0.079848
Stroke           1.000000     -0.079985  0.176599  0.192266  0.085944
PhysActivity    -0.079985      1.000000 -0.238027 -0.276868 -0.082939
GenHlth          0.176599     -0.238027  1.000000  0.458363  0.115068
DiffWalk         0.192266     -0.276868  0.458363  1.000000  0.166384
Age              0.085944     -0.082939  0.115068  0.166384  1.000000
Diabetes_risk    0.125427     -0.158666  0.306559  0.272646  0.249054

                 Diabetes_risk
Diabetes_binary       1.000000
HighBP                0.381516
HighChol              0.289213
CholCheck             0.115382
BMI                   0.238733
Stroke                0.125427
PhysActivity         -0.158666
GenHlth               0.306559
DiffWalk              0.272646
Age                   0.249054
Diabetes_risk         1.000000
```


Correlation Matrix
Diabetes_binary 1.00 0.38 0.29 0.12 0.24 0.13 -0.16 0.31 0.27 0.25 1.00
HighBP 0.38 1.00 0.32 0.10 0.22 0.13 -0.14 0.23 0.23 0.29 0.38
HighChol 0.29 0.32 1.00 0.09 0.15 0.10 -0.09 0.18 0.16 0.23 0.29
CholCheck 0.12 0.10 0.09 1.00 0.05 0.02 -0.01 0.04 0.04 0.09 0.12
BMI 0.24 0.22 0.15 0.05 1.00 0.02 -0.10 0.09 0.12 0.08 0.24
Stroke 0.13 0.13 0.10 0.02 0.02 1.00 -0.08 0.18 0.19 0.09 0.13
PhysActivity -0.16 -0.14 -0.09 -0.01 -0.10 -0.08 1.00 -0.24 -0.28 -0.08 -0.16
GenHlth 0.31 0.23 0.18 0.04 0.09 0.18 -0.24 1.00 0.46 0.12 0.31
DiffWalk 0.27 0.23 0.16 0.04 0.12 0.19 -0.28 0.46 1.00 0.17 0.27
Age 0.25 0.29 0.23 0.09 0.08 0.09 -0.08 0.12 0.17 1.00 0.25
Diabetes_risk 1.00 0.38 0.29 0.12 0.24 0.13 -0.16 0.31 0.27 0.25 1.00
Diabetes_binary
HighBP
HighChol
CholCheck
BMI
Stroke
PhysActivity
GenHlth
DiffWalk
Age
Diabetes_risk
1.0
0.8
0.6
0.4
0.2
0.0
−0.2



#5. # KDD Goals Risk factors predictive of diabetes included in this mode: BMI, HighChol, highBp, PhysActivity, AnyHealthcare All of these factors are intertwined with eachother to make up a persons general health. e.g not having healthcare may lead to a person not knowing whether or not they have highBP or highChol. OR not being physically active may also lead to both

1 possible new features

A 'helath_risk' feature which will take into account BMI. HighChol, HighBP, and see if these values are in the 1 category and if so, it would be considered a health risk

possibly create different categories of BMI like the WHO has 'Underweight: 0 - 18.5', 'Normal: 18.6 - 24.9', 'overweight: 25 - 40', 'Obese: 40+'

If a person has no healthcare they are at higher risk since they will not know how helathy/unhealthy they are

[49]:
```
# create a health_risk_score based on features that may have an effect
data['Health_risk_score'] = data['BMI'] + data['HighChol'] + data['HighBP'] +␣
 ↪data['PhysActivity'] + data['PhysActivity']

# BMI categories described above
data['BMI_by_category'] = pd.cut(data['BMI'],
```

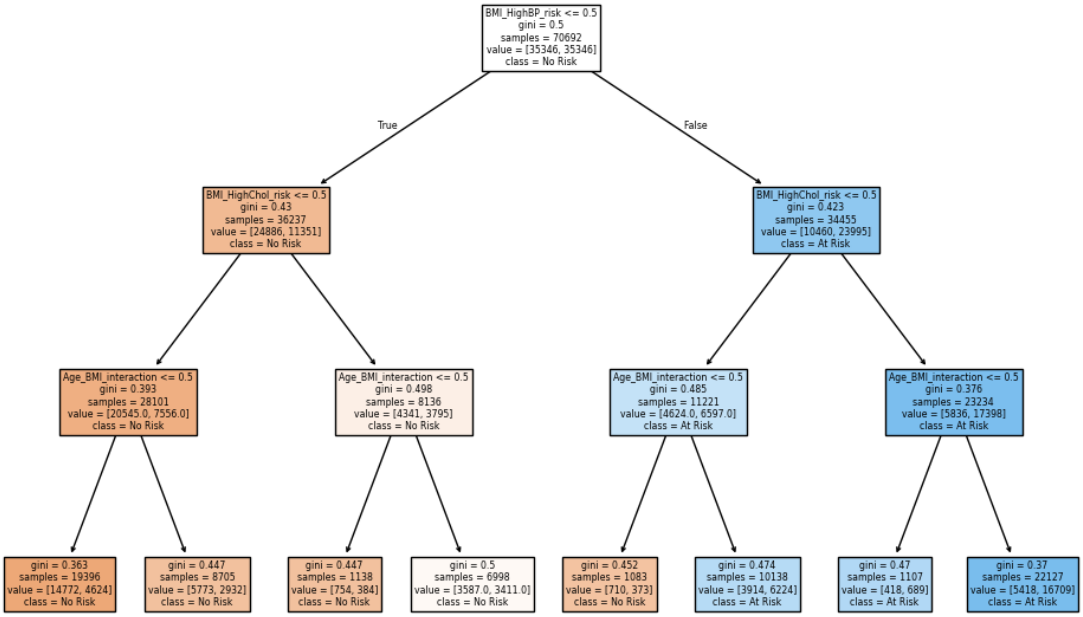
```
                      bins=[0, 18.5, 24.9, 29.9, 40],
                      labels=['Underweight', 'Normal', 'Overweight', 'Obese'])

# impact that PhysActivity may have on BMI
data['BMI_PhysActivity'] = data['BMI'] * data['PhysActivity']

# Impact of age
data['Age_BMI_interaction'] = data['Age'] * data['BMI']
```

#6. #Search for patterns of interest As stated above, Potential patterns of interest can be: A person who has a high BMI (> 25), has a highre risk of having diabetes. A person who has High Blood Pressure (1) has a higher risk of habing diabetes. Someone who has the combination of both will have a higher chance of having diabetes. A person who has high cholesterol will have a higher chance of having diabetes.

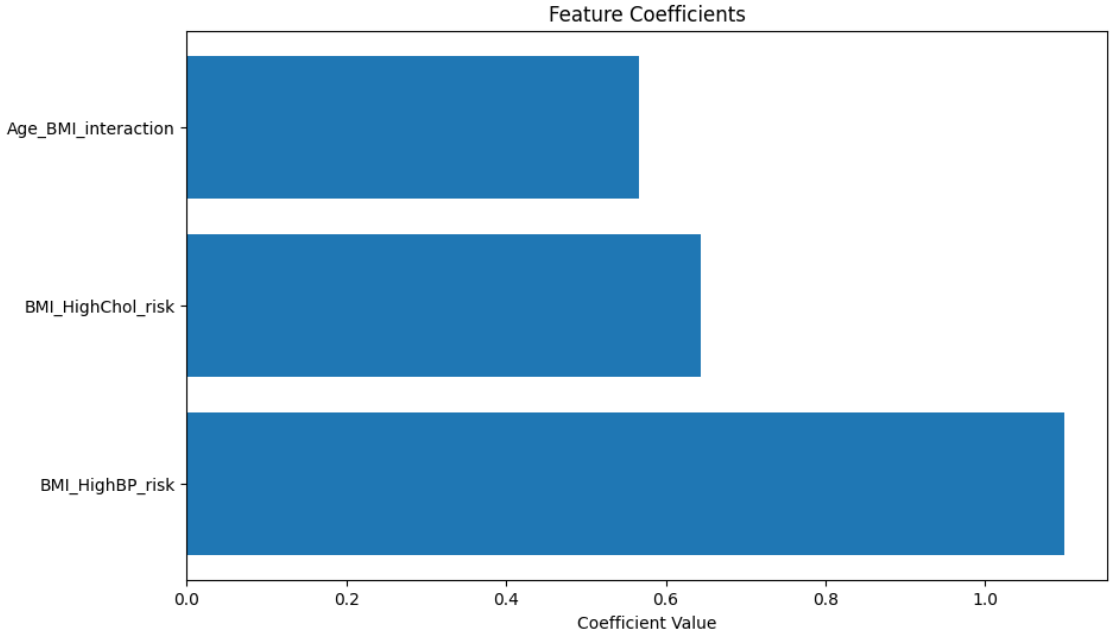
[50]:
```
# Ensure binary columns are integers
data['BMI'] = data['BMI'].astype(int)
data['HighBP'] = data['HighBP'].astype(int)
data['HighChol'] = data['HighChol'].astype(int)
# Got code above from chatGPT, was having Type error problems. made sure␣
 ↪features where of type int


# establish the risk of having BMI > 25 and high blood pressure (1)
data['BMI_HighBP_risk'] = data['BMI'] & data['HighBP']

# establish the risk of having BMI > 25 and having high Cholesterol (1)
data['BMI_HighChol_risk'] = data['BMI'] & data['HighChol']

# Risk of all three
#data['BMI_HighBP_HighChol_risk'] = data['BMI'] & data['HighBP'] &␣
 ↪data['HighChol']

# in order to determine the risk : use decision tree

# features : BMI_HighBP_risk, BMI_HighChol_risk, BMI, HighChol, HighBP
X = data[['BMI_HighBP_risk', 'BMI_HighChol_risk', 'Age_BMI_interaction']] #␣
 ↪Removed ['BMI', 'HighChol', 'HighBP', 'BMI_HighBP_HighChol_risk'] to focus␣
 ↪on the pattern
#target variable is Diabetes_risk
y = data['Diabetes_risk']

# decision tree
dt = DecisionTreeClassifier(max_depth=5, random_state=42)
dt.fit(X, y)

print(data['Diabetes_risk'].value_counts())
```

```
# Visualize the decision tree
plt.figure(figsize=(12, 8))
plot_tree(dt, feature_names=X.columns, class_names=['No Risk', 'At Risk'],␣
 ↪filled=True)
plt.show()
plt.close()


# using Regression

# going to use same features and target variable

lr = LogisticRegression(max_iter = 1000)
lr.fit(X,y)

# esatblish coefficients
coefs = lr.coef_[0] # get the coefficients as a 1D array

#print intercept and coefficients
print("Intercept:", lr.intercept_)
print("Coefficients:", coefs)

#predict the probability of the class being 1 (pre-diabetic/diabetic)
prob_pred = lr.predict_proba(X)[:,1]
# predict the class
class_pred = lr.predict(X)
print("Predictions (Classes):", class_pred)

# Create a bar chart to visualize the coefficients
features = X.columns  # Get feature names
plt.figure(figsize=(10, 6))
plt.barh(features, coefs)
plt.xlabel('Coefficient Value')
plt.title('Feature Coefficients')
plt.show()
```

```
Diabetes_risk
0    35346
1    35346
Name: count, dtype: int64
```




```
Intercept: [-1.21151933]
Coefficients: [1.09880581 0.64395017 0.56576438]
Predictions (Classes): [0 1 0 … 1 0 1]
```

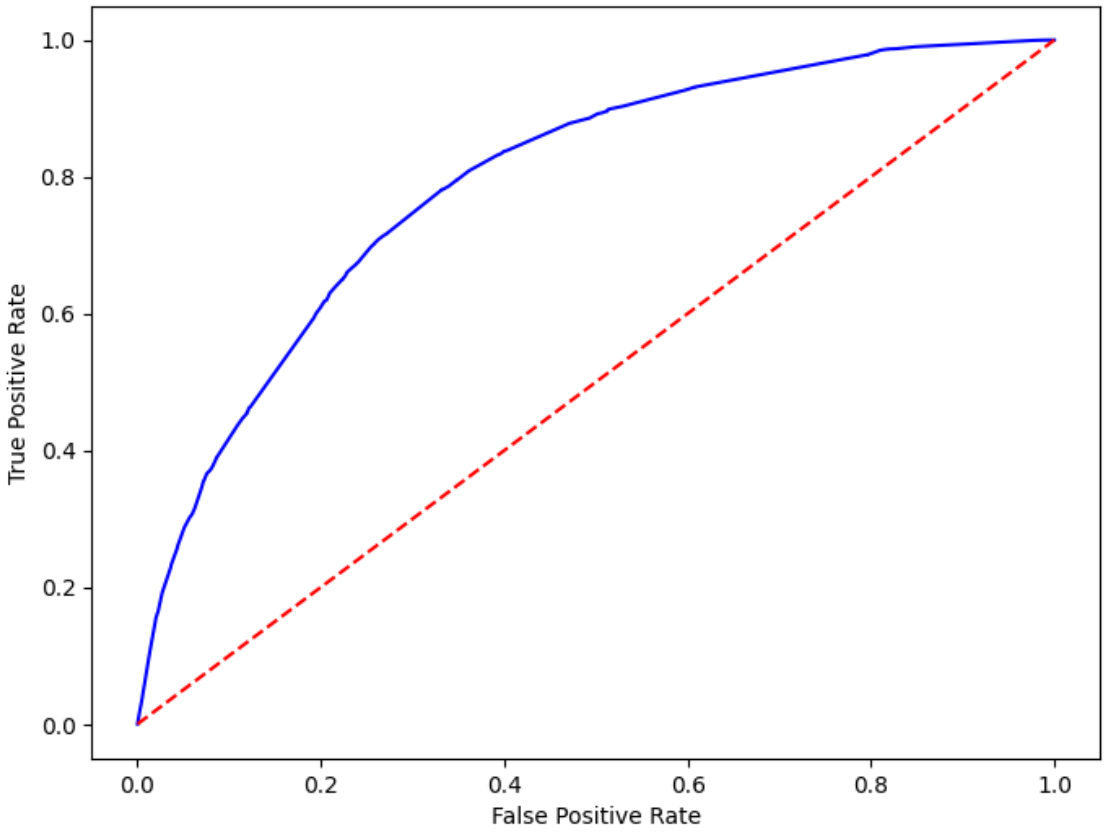


#7. The first supervised machine learning model that I will be using is Logisitic Regression. I think that this model would be uswful because the features are in a binary format and a logistic regression model would be useful in finding linea correaltions I stated above like higher BMI leading to an increased risk in having diabetes.

[51]:
```
# Features
X_train = data[[ 'HighBP', 'HighChol', 'PhysActivity', 'DiffWalk' ,'CholCheck',␣
 ↪'Stroke', 'GenHlth', 'BMI_HighBP_risk', 'BMI_HighChol_risk', 'Age']] #␣
 ↪Removed ['BMI', 'HighChol', 'HighBP', 'BMI_HighBP_HighChol_risk

# adding BMI ? throws off the BMI_HIghBP and BMI_CholCheck

# target will the the risk of diabetes
y_train = data['Diabetes_risk']

# going to split the data into 80% for training and 20% for testing
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.
 ↪2, random_state=42)

lr_model = LogisticRegression(C=1.0, penalty='l2', solver='liblinear',␣
 ↪max_iter=1000, random_state=42)
lr_model.fit(X_train, y_train)

# Evaluate the model on the validation set
val_accuracy = lr_model.score(X_val, y_val)
print(f"Logistic Regression Validation Accuracy: {val_accuracy:.2f}")

# test the model
y_pred = lr_model.predict(X_val)

# Evaluate the model
accuracy = accuracy_score(y_val, y_pred)
print(f"Logistic Regression Accuracy: {accuracy:.2f}")


print("Logistic Regression Coefficients:", lr_model.coef_)

#ROC
y_proba = lr_model.predict_proba(X_val)[:, 1]

# Calculate ROC curve
fpr, tpr, thresholds = roc_curve(y_val, y_proba)

# Calculate AUC
roc_auc = roc_auc_score(y_val, y_proba)
```

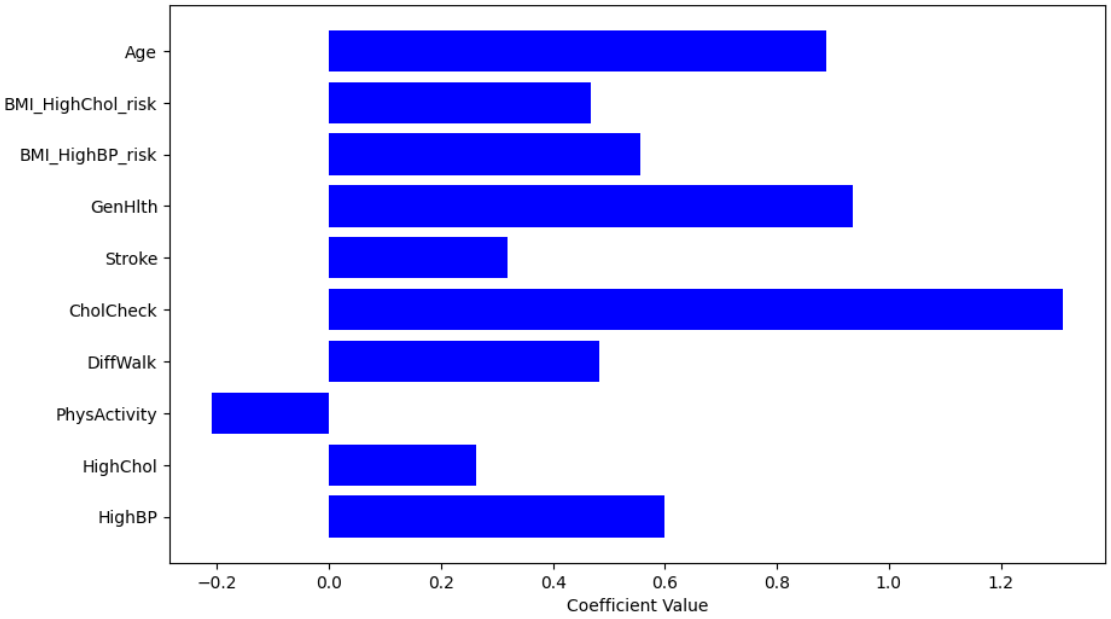
```python
# Plot the ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', label=f"ROC Curve (AUC = {roc_auc:.2f})")
plt.plot([0, 1], [0, 1], color='red', linestyle='--', label="Random Guess")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.show()
plt.close()

print(f"AUC Score: {roc_auc:.2f}")

# visualize the how the coefficeints effect risk
features = X_train.columns
coefficients = lr_model.coef_[0]
plt.figure(figsize=(10, 6))
plt.barh(features, coefficients, color='blue')
plt.xlabel('Coefficient Value')
```

```
Logistic Regression Validation Accuracy: 0.72
Logistic Regression Accuracy: 0.72
Logistic Regression Coefficients: [[ 0.59991217  0.26208823 -0.20911287
0.48187822  1.31049931  0.31824876
   0.93501237  0.55686604  0.46747222  0.88934589]]
```



```
AUC Score: 0.79
```

[51]:
```
Text(0.5, 0, 'Coefficient Value')
```



2 Assessment of Logistic Regression

The logistic regression model has an accuracy of 71%. The features I expected to have an impact on whether or not an individual is at risk of having diabetes/having diabetes where on point. The 'physActivity' feature had a negative correlation showing that getting in physical activity that does not include your job can be helpful in lowering an individuals chance of getting diabetes.

The AUC score was 0.80 meaning this logistic regression model is doing a good job in making accurate predictions between the positive and negative classes.

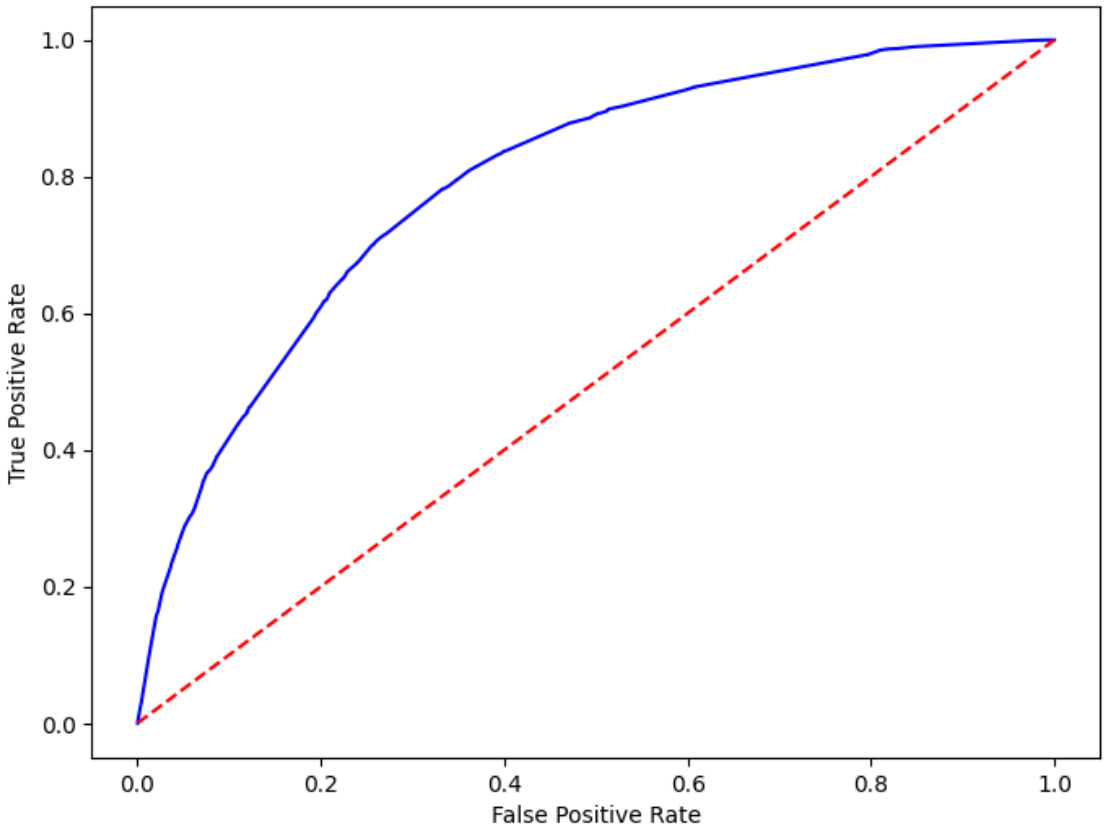
3 SVM

The next supervised machine learning model I will be using is the SVM.

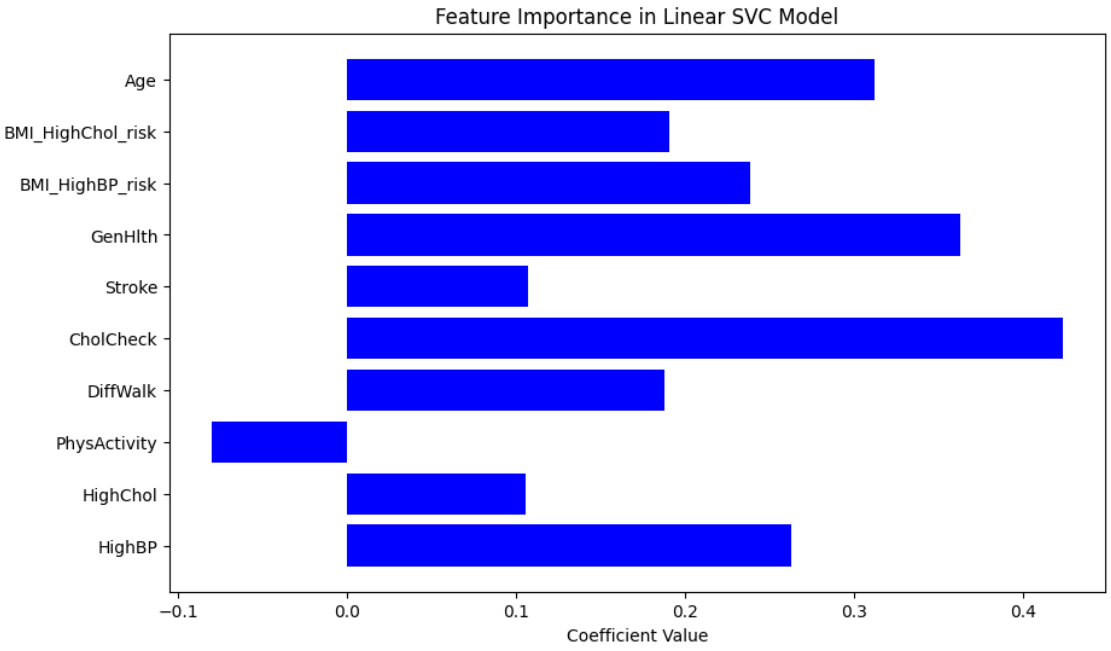
[52]:
```
# Features and target variable will remain the same

lsvc_model = LinearSVC(C=1.0, random_state=42)
lsvc_model.fit(X_train, y_train)

# Predict the labels for the validation set
```

```
y_pred_lsvc = lsvc_model.predict(X_val)

# Calculate accuracy
accuracy_svc = accuracy_score(y_val, y_pred_lsvc)
print(f"LinearSVC Accuracy: {accuracy_svc:.2f}")

# Get decision function values (distance from the decision boundary)
y_decision_svc = lsvc_model.decision_function(X_val)

# Calculate AUC score
auc_score_svc = roc_auc_score(y_val, y_decision_svc)
print(f"LinearSVC AUC Score: {auc_score_svc:.2f}")


# Plot the ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', label=f"ROC Curve (AUC = {auc_score_svc:.2f})")
plt.plot([0, 1], [0, 1], color='red', linestyle='--', label="Random Guess")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.show()
plt.close()
print("LinearSVC Feature Weights:", lsvc_model.coef_)

# Plot coefficients
features = X_train.columns
coefficients = lsvc_model.coef_[0]
plt.figure(figsize=(10, 6))
plt.barh(features, coefficients, color='blue')
plt.xlabel('Coefficient Value')
plt.title('Feature Importance in Linear SVC Model')
plt.show()
plt.close()
```

```
LinearSVC Accuracy: 0.72
LinearSVC AUC Score: 0.79
```




```
LinearSVC Feature Weights: [[ 0.26343888  0.1058462  -0.07976294  0.1877539
0.4236935   0.1069695
   0.36318178  0.2389068   0.19086376  0.31249272]]
```



Analayzing the result from LinearSVC, I see that it performed the same as the logisitc regression model. one possible reason for this is because the data is setup in a linear fashion, meaning that almost all of the features I am testing for increase the chance of someone having Diabetes if the healthmarkers are elevated.

4 Linear Regression

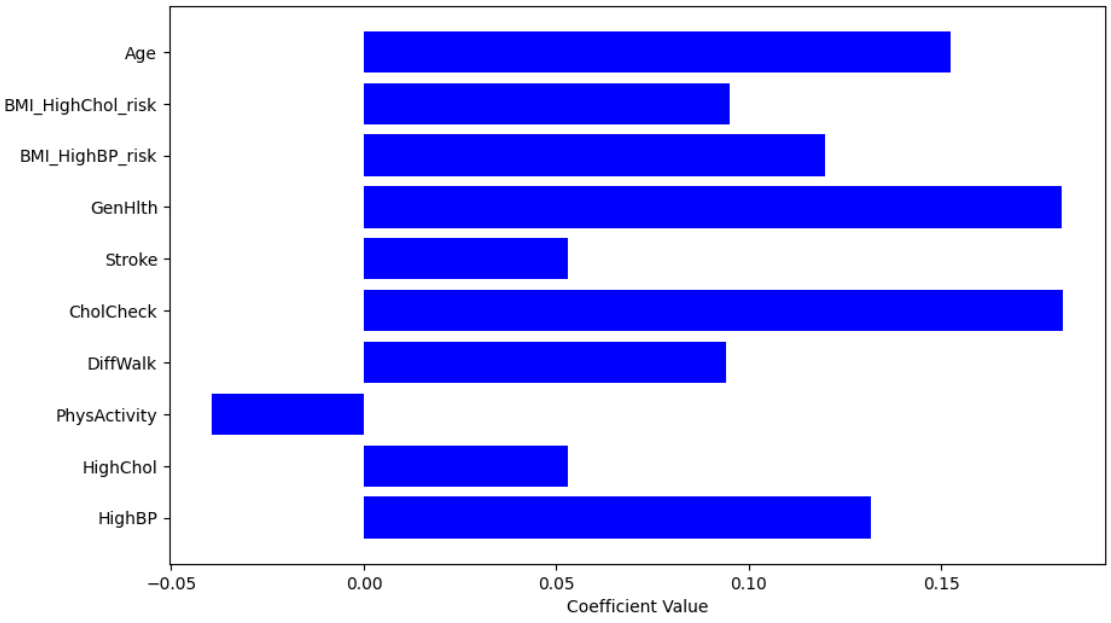
The next model I will be using is linear regression. I will be using linear regression beacause I think that the features havea linear relationship with the target varibale. This means that as the helath markers that I am testing for increase, the likelihood of a person having diabetes will also increase. Due to this relationship I think that this model wil perform the best compared to the other model above.

[53]:
```
# Features and target variable will remain the same

# Linear regression model
linreg_model = LinearRegression()
linreg_model.fit(X_train, y_train)

# Predict the labels for the validation set
y_pred_linreg = linreg_model.predict(X_val)

# Calculate mean squared
mse_linreg = mean_squared_error(y_val, y_pred_linreg)
print(f"Linear Regression Mean Squared Error: {mse_linreg:.2f}")
```

```
# r2 score
r2_score = linreg_model.score(X_val, y_val)
print(f"Linear Regression R^2 Score: {r2_score:.2f}")

# get the coefficients of the model
print("Linear Regression Coefficients:", linreg_model.coef_)

#visualie the coefficients
features = X_train.columns
coefficients = linreg_model.coef_
plt.figure(figsize=(10, 6))
plt.barh(features, coefficients, color='blue')
plt.xlabel('Coefficient Value')
```

```
Linear Regression Mean Squared Error: 0.19
Linear Regression R^2 Score: 0.26
Linear Regression Coefficients: [ 0.13166081  0.05303413 -0.03943257  0.09399055
0.18149577  0.05291477
  0.18124339  0.11968532  0.09516584  0.1523396 ]
```

[53]:
```
Text(0.5, 0, 'Coefficient Value')
```



[54]:
```
# Install necessary tools
!apt-get install texlive-xetex texlive-fonts-recommended texlive-plain-generic␣
↪pandoc
```

```
# Upload the .ipynb file
from google.colab import files
f = files.upload()

# Convert ipynb to PDF
import subprocess
import os

# Get the filename of the uploaded file
file0 = list(f.keys())[0]
print(f"Uploaded file: {file0}")

# Install nbconvert if not already installed
!pip install nbconvert

# Convert the ipynb file to pdf using nbconvert
output_file = "/content/" + file0[:-5] + ".pdf"

# Run nbconvert to generate the PDF
_ = subprocess.run(["jupyter", "nbconvert", "--to", "pdf", "--output",␣
 ↪output_file, "/content/" + file0])

# Check if the file was generated
if os.path.exists(output_file):
    print(f"PDF file generated successfully: {output_file}")
    files.download(output_file)  # Download the PDF
else:
    print("Error: PDF file not found.")
```

```
E: dpkg was interrupted, you must manually run 'dpkg --configure -a' to correct
the problem.
```

```
<IPython.core.display.HTML object>
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-54-6f80466a4b86> in <cell line: 6>()
      4 # Upload the .ipynb file
      5 from google.colab import files
----> 6 f = files.upload()
      7
      8 # Convert ipynb to PDF

/usr/local/lib/python3.10/dist-packages/google/colab/files.py in␣
 ↪upload(target_dir)
     70   """
     71
---> 72   uploaded_files = _upload_files(multiple=True)
```

```
     73   # Mapping from original filename to filename as saved locally.
     74   local_filenames = dict()

/usr/local/lib/python3.10/dist-packages/google/colab/files.py in␣
 ↪_upload_files(multiple)
    162
    163   # First result is always an indication that the file picker has␣
 ↪completed.
--> 164   result = _output.eval_js(
    165       'google.colab._files._uploadFiles("{input_id}", "{output_id}")'.
 ↪format(
    166           input_id=input_id, output_id=output_id

/usr/local/lib/python3.10/dist-packages/google/colab/output/_js.py in␣
 ↪eval_js(script, ignore_result, timeout_sec)
     38   if ignore_result:
     39     return
---> 40   return _message.read_reply_from_input(request_id, timeout_sec)
     41
     42

/usr/local/lib/python3.10/dist-packages/google/colab/_message.py in␣
 ↪read_reply_from_input(message_id, timeout_sec)
     94     reply = _read_next_input_message()
     95     if reply == _NOT_READY or not isinstance(reply, dict):
---> 96       time.sleep(0.025)
     97       continue
     98     if (

KeyboardInterrupt:
```