Main objective of the analysis:

- The main objective of my analysis will be concerning diabetes. It will be a prediction focused on determining if someone is pre-diabetic based on attributes.
- People who will benefit from this are those individuals who are pre-diabetic. People who are pre- Type 2 diabetes who know their condition beforehand can take measures to prevent this from occurring, which will save them medical expenses in the long run.
- Also, Insurance companies can also benefit because when their customers get diabetes, they will have to cover some of their expenses. Of course, if their customer never gets diabetes, they won't have to be paying these expenses.

Brief description of the data set:

- Dataset was obtained from kaggle and rated 10/10 usability. https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset?resource=download
- The CSV I used has just over 76,000 rows of data. I used the 50-50 class split dataset which downsampled from the original dataset, where there were a lot more non-diabetics than pre-diabetics which really messed up my model.
- In this analysis, I will be using all 21 features in the dataset(BMI, Smoking, Blood Pressure, etc.) in order to create a model that can **maximize the recall function**. I chose this error metric because we really want to punish False-Negatives, or cases where we misidentify a pre-diabetic.

Brief summary of data exploration and actions taken for data cleaning and feature engineering:

- Some columns are continuous, whereas some are binary. To solve this, I used the standardScaler library to scale the models for logistic regression and Support Vector Machines
- The feature of interest, the 'Diabetes' column,actually has 3 values. 0 (non-diabetic), 1(pre-diabetic), and 2(diabetic). Because the analysis is focused on predicting pre-diabetes only, I dropped rows identifying already diabetic individuals. Then, as stated earlier, I downsampled so that there were an equal number of these two classes.

Summary of training at least three different classifier models:

- 1. Logistic Regression
  - ★ Scaled using StandardScaler
  - ★ Split using train_test_split with a test size of .3
  - ★ Penalty function was L2 (Ridge). I chose this because there weren't too many features and there is a possibility that all of them are important.
  - ★ 5 different folds for cross validation
  - ★ C = 10, meaning there is weak regularization. This is because I figured that all data between train and test data would be similar enough where strong regularization wouldn't really make a difference.
  - ★ **Coefficients:**
    - **Feature | Coefficient**
    - **==========**
    - **0 HighBP 0.746872**
    - **1 HighChol 0.588085**
    - **2 CholCheck 1.318270**
    - **3 BMI 6.368114**
    - **4 Smoker -0.012894**
    - **5 Stroke 0.163321**
    - **6 HeartDiseaseorAttack 0.264393**
    - **7 PhysActivity -0.033155**
    - **8 Fruits -0.023658**
    - **9 Veggies -0.068416**
    - **10 HvyAlcoholConsump -0.747809**
    - **11 AnyHealthcare 0.049747**
    - **12 NoDocbcCost 0.066541**
    - **13 GenHlth 2.363298**
    - **14 MentHlth -0.171066**
    - **15 PhysHlth -0.222826**
    - **16 DiffWalk 0.096318**
    - **17 Sex 0.242714**
    - **18 Age 1.815969**
    - **19 Education -0.242375**
    - **20 Income -0.353070**
    -
    -
  - ★ **Final Recall Score: 0.757609710550887**

- **2. Support Vector Machines**
  - ★ Scaled using StandardScaler
  - ★ No splitting, all data was used as the training data
  - ★ **LinearSVC used**
  - ★ **Final Recall Score: 0.7735528772704124**

- **3. Random Forests**
  - ★ Data not scaled because it doesn't matter for tree-based models
  - ★ Split using train_test_split with a test size of .3
  - ★ Performed using GrisSearchCV in order to ensure optimal hyperparameter tuning
  - ★ 'n_estimators': [2*n+1 for n in range(5)]
  - ★ 'max_depth' : [2*n+1 for n in range(7)]
  - ★ (Anything more than the two parameters above wouldn’t work, as my computer couldn't handle it)
  - ★ Scoring was recall
  - ★ **Best Parameters: {'max_depth': 7, 'n_estimators': 9}**
  - ★ **Final Recall Score: 0.7903678881542111**

Final Recommendation:

Out of the three models listed above, I recommend the Random Forest model. This is because it was the highest **Recall** score of the three models. Even though it is somewhat of a black-box model in that we cannot see everything occurring behind the scenes, that is not so important as correctly classifying individuals. This is not a problem because we arent presenting our findings to a business or any corporation that requires extreme trust. If we really wanted to, we could use a LIME to simplify the model in a way that anyone could interpret. If no one would be willing to acknowledge the model without a thorough understanding of the calculations made behind the scene, then I would probably recommend Logistic Regression due to it being a self-interpretable model.

Summary Key Findings and Insights:

- From the logistic regression model, **3 features stand out in terms of importance due to their large coefficients:**
  - ★ **#1: BMI** (This makes a lot of sense!)
  - ★ **#2: GenHlth** (people rated what they thought their health was from 1-5), again this makes a lot of sense
  - ★ **#3: Age** (Older = more likely to be pre-diabetic). I actually had never thought about this.
- **The highest recall (.79 from Random Forests) was lower than I thought it would be.** To me, this is interesting because it might suggest that there is a lot of variation in the features for those who are pre-diabetic.

If this is true, it would disprove my belief earlier that they would all be really similar!

Suggestions for next steps in analyzing this data:

- The next thing I would do to analyze this data would be to create a LIME for the random forest. By slightly tweaking inputs, it could generate a sample of one of the main decision trees that was used to make the original model.
- Another thing I would do is implement non-linear kernels for the SVM model. This can capture more complex decision boundaries that could in ruth make the model better and increase the recall
- Thirdly, we could show the ROC curve for each model rather than just the recall if we also cared about the False-Positive rate. But since I thought that a False-Negative was far more detrimental than a False-Positive, I didn't do this.

==========

Code used:

```
from sklearn.linear_model import LogisticRegressionCV
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score as precision
from sklearn.metrics import recall_score as recall
from sklearn.metrics import confusion_matrix,
accuracy_score, roc_auc_score
from sklearn.preprocessing import label_binarize
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import GridSearchCV
import numpy as np
import pandas as pd
from sklearn.svm import LinearSVC
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('diabetes_coursera.csv')
```

```
#########################################################
#################

X = df.drop(columns='Diabetes_binary')
Y = df['Diabetes_binary']
scalerLR = MinMaxScaler()
X_lr_standardized = scalerLR.fit_transform(X)
X_train, X_test, y_train, y_test =
train_test_split(X_lr_standardized,Y,test_size=.3)

lr = LogisticRegression(solver='liblinear').fit(X_train,
y_train)
lr_l1 = LogisticRegressionCV(Cs=10, cv=5, penalty='l2',
solver='liblinear').fit(X_train, y_train)
coef_df = pd.DataFrame({
    'Feature': X.columns,
    'Coefficient': lr_l1.coef_[0]
})

# Display the coefficients
print(coef_df)
y_pred = lr_l1.predict(X_test)

recall_lr_l1 = recall(y_test,y_pred)

print(recall_lr_l1)

#############################################

X_svm = df.drop(columns='Diabetes_binary')
scaler = MinMaxScaler()
```

```
X_svm_standardized = scaler.fit_transform(X_svm)
X_svm_standardized = pd.DataFrame(X_svm_standardized,
columns=X_svm.columns)
print(type(X_svm_standardized))
y_svm = df['Diabetes_binary']
LSVC = LinearSVC()
LSVC.fit(X_svm_standardized, y_svm)


y_pred_svm = LSVC.predict(X_svm_standardized)
recall_svm = recall(y_svm,y_pred_svm)
print(recall_svm)

##########################################################
#####

X_rf = df.drop(columns='Diabetes_binary')
y_rf = df['Diabetes_binary']
print(sum(y_rf==1)/y_rf.shape[0])
X_train_rf, X_test_rf, y_train_rf, y_test_rf =
train_test_split( X_rf, y_rf, test_size=0.2,
random_state=4)
rf = RandomForestClassifier()
rf.get_params().keys()

param_grid = {'n_estimators': [2*n+1 for n in range(5)],
              'max_depth' : [2*n+1 for n in range(7) ],
              }
search = GridSearchCV(estimator=rf,
param_grid=param_grid,scoring='recall')
search.fit(X_train_rf, y_train_rf)
```

```
print("Best Parameters:", search.best_params_)
print("Best recall Score:", search.best_score_)
```