Heart Attack Risk Prediction

PRESENTED BY: RASHI AGRAWAL, HUNTER DAVIS, JENIQUE FAHIE AND DANIEL O'MEARA

Features:
Patient ID | Age
Sex | Cholesterol
Blood Pressure | Heart Rate
Diabetes | Family History
Smoking | Obesity
Alcohol Consumption | Exercise Hours Per Week
Diet | Previous Heart Problems
Medication Use | Stress Level
Sedentary Hours Per Day | Income
BMI | Triglycerides
Physical Activity Days Per Week | Sleep Hours Per Day
Country | Continent
Hemisphere

Our Dataset

HEART ATTACK RISK PREDICTION

This is a synthetic AI-generated dataset sourced from Kaggle.com. (https://www.kaggle.com/datasets/iamsouravbanerjee/heart-attack-prediction-dataset?resource=download)

The target variable we are trying to predict is **Heart Attack Risk**.

Contains:
- 8,763 rows
- 26 columns

```
# IS target column balanced?
df['Heart Attack Risk'].value_counts()
```

```
Heart Attack Risk
0    5624
1    3139
```

64.2%
35.8%

Blood Pressure	Heart Rate	Diabetes	Family History	Smoking	Obesity	Alcohol Consumption	Exercise Hours Per Week	Diet	Previous Heart Problems	Medication Use	Stress Level	Sedentary Hours Per Day	Income	BMI	Triglycerides	Physical Activity Days Per Week	Sleep Hours Per Day	Country	Continent	Hemisphere	Heart Attack Risk
141/85	101	No	No	0	1	Light	14.744881	Unhealthy	0	1	4	10.922177	94152	30.589796	374	3	4	Nigeria	Africa	Northern Hemisphere	1
137/82	89	Yes	No	0	1	Light	16.228489	Unhealthy	0	1	7	1.208610	159792	31.584511	678	0	8	Australia	Australia	Southern Hemisphere	1
138/93	86	Yes	Yes	1	0	None	6.818887	Average	0	1	4	9.514556	254952	34.711478	736	1	5	Argentina	South America	Southern Hemisphere	1
132/94	109	Yes	No	1	1	Light	18.297860	Average	1	1	6	9.015221	25229	29.022289	152	2	5	Spain	Europe	Southern Hemisphere	1
91/89	84	Yes	Yes	1	0	Moderate	10.980701	Average	0	1	4	10.020410	229179	35.966244	744	5	8	Japan	Asia	Northern Hemisphere	1

Processing the Data

Dropped unnecessary columns:

- Patient ID, Country, Continent, Hemisphere, Income

Reduced the number of unique values in several columns:

- Exercise Hours Per Week, Sedentary Hours Per Day, BMI, Blood Pressure
- Split Blood Pressure values (e.g., 120/80) into two separate columns.
- Converted remaining floating-point numbers to integers.

Converted categorical columns to numeric:

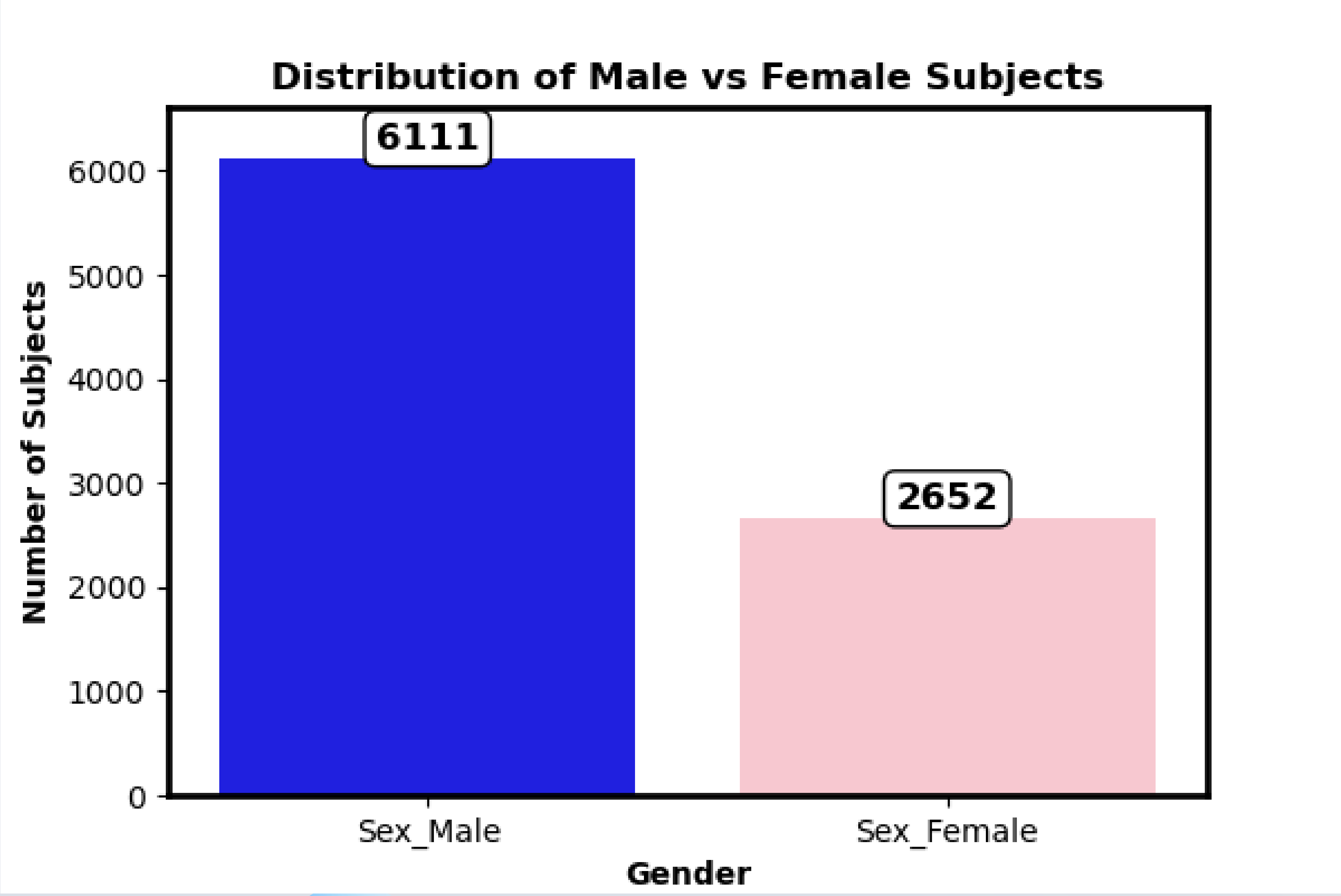
- Used `get_dummies()` to encode categorical columns (e.g., Diet: Average, Healthy, Unhealthy), creating binary columns.

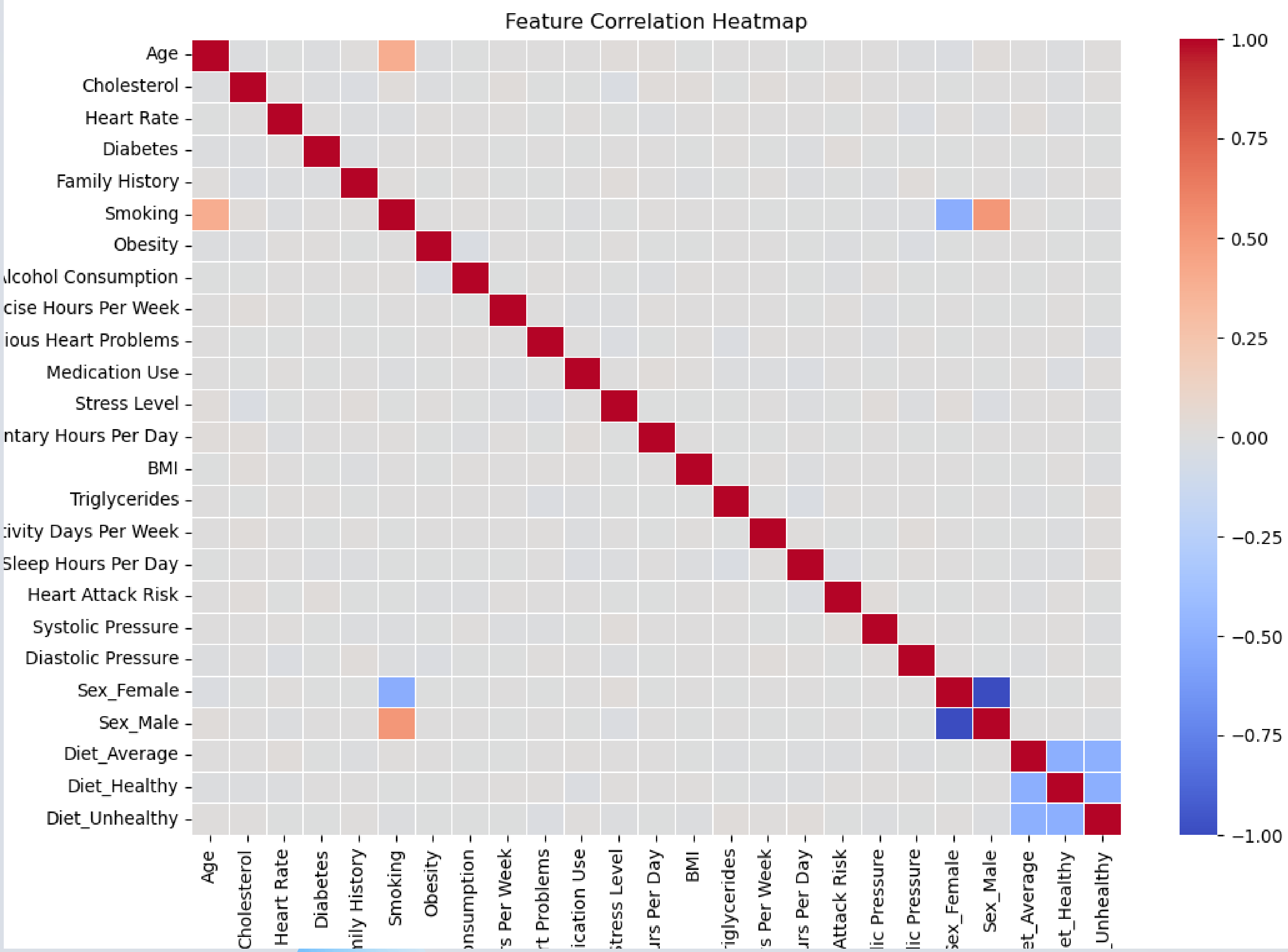
Exported cleaned Data Frame into an SQLite Database

```
df.nunique()
```

```
Patient ID                          8763
Age                                   73
Sex                                    2
Cholesterol                          281
Blood Pressure                      3915
Heart Rate                            71
Diabetes                               2
Family History                         2
Smoking                                2
Obesity                                2
Alcohol Consumption                    2
Exercise Hours Per Week             8763
Diet                                   3
Previous Heart Problems                2
Medication Use                         2
Stress Level                          10
Sedentary Hours Per Day             8763
Income                              8615
BMI                                 8763
Triglycerides                        771
Physical Activity Days Per Week        8
Sleep Hours Per Day                    7
Country                               20
Continent                              6
Hemisphere                             2
Heart Attack Risk                      2
```

```
Age                                   73
Cholesterol                          281
Heart Rate                            71
Diabetes                               2
Family History                         2
Smoking                                2
Obesity                                2
Alcohol Consumption                    2
Exercise Hours Per Week               20
Previous Heart Problems                2
Medication Use                         2
Stress Level                          10
Sedentary Hours Per Day               12
BMI                                   22
Triglycerides                        771
Physical Activity Days Per Week        8
Sleep Hours Per Day                    7
Heart Attack Risk                      2
Systolic Pressure                     91
Diastolic Pressure                    51
Sex_Female                             2
Sex_Male                               2
Diet_Average                           2
Diet_Healthy                           2
Diet_Unhealthy                         2
```


Distribution of Male vs Female Subjects
6111
2652
Number of Subjects
6000
5000
4000
3000
2000
1000
0
Sex_Male
Sex_Female
Gender



Feature Correlation Heatmap
Age
Cholesterol
Heart Rate
Diabetes
Family History
Smoking
Obesity
lcohol Consumption
cise Hours Per Week
ious Heart Problems
Medication Use
Stress Level
ntary Hours Per Day
BMI
Triglycerides
tivity Days Per Week
Sleep Hours Per Day
Heart Attack Risk
Systolic Pressure
Diastolic Pressure
Sex_Female
Sex_Male
Diet_Average
Diet_Healthy
Diet_Unhealthy
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

Method 1: Logistic Regression

Initial Model Performance – Scaled :

- Accuracy: 0.53
- Takeaway: Low recall for Class 1 (Risk Present) suggests imbalance

Logistic Regression Accuracy: 0.53

Classification Report:

	precision	recall	f1-score	support
0	0.64	0.61	0.63	1406
1	0.35	0.38	0.37	785
accuracy			0.53	2191
macro avg	0.50	0.50	0.50	2191
weighted avg	0.54	0.53	0.53	2191

Tuning Method 1: SMOTE/Over Sampling

- Goal: Address class imbalance by oversampling the minority class (1)
- Effect on Accuracy: No improvement (remained at 0.53)
- Takeaway: SMOTE alone did not improve model performance, indicating imbalance may not be the primary issue

Logistic Regression Accuracy after SMOTE: 0.53
Classification Report:

	precision	recall	f1-score	support
0	0.64	0.61	0.63	1406
1	0.35	0.38	0.37	785
accuracy			0.53	2191
macro avg	0.50	0.50	0.50	2191
weighted avg	0.54	0.53	0.53	2191

Tuning Method 2: Feature Selection (RFE)

- Approach: Retained only important features based on RFE
- Effect on Accuracy: Improved to 0.64
- Takeaway: Higher accuracy but poor ability to predict minority class
    - Model became too biased towards Class 0, suggesting feature selection removed key predictors for Class 1

Accuracy after feature selection: 0.64
Classification Report:

	precision	recall	f1-score	support
0	0.64	1.00	0.78	1406
1	0.00	0.00	0.00	785
accuracy			0.64	2191
macro avg	0.32	0.50	0.39	2191
weighted avg	0.41	0.64	0.50	2191

Method 2: Random Forrest

Each decision tree in the Random Forest is trained on a subset of the data and makes predictions independently. For classification, the final prediction is based on majority voting among the trees, while for regression, the predictions are averaged.

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

# Initialize the Random Forest model ( good for classification tasks)
model = RandomForestClassifier(n_estimators=100, random_state=42)
```

```
Accuracy: 0.64
```

Steps Taken to Improve Random Forest Model

```python
# List of n_estimators values to experiment with
n_estimators_values = [100, 500, 1000]

for n_estimators in n_estimators_values:
    print(f"Training model with n_estimators={n_estimators}")

    # Initialize the Random Forest model with different n_estimators values
    model = RandomForestClassifier(
        n_estimators=n_estimators,
        max_features='sqrt',
        random_state=42
    )
```

```
Training model with n_estimators=100
Accuracy: 0.64
```

```
Training model with n_estimators=500
Accuracy: 0.64
```

```
Training model with n_estimators=1000
Accuracy: 0.64
```

```python
# Experiment with different values for max_features
for max_features in ['sqrt', 'log2', 0.5, 1, None]:
    print(f"Training model with max_features={max_features}")

    # Initialize the Random Forest model with different max_features
    model = RandomForestClassifier(n_estimators=100, max_features=max_features, random_state=42)
```

```
Training model with max_features=0.5
Accuracy: 0.63
```

```
Training model with max_features=1
Accuracy: 0.64
```

```
Training model with max_features=sqrt
Accuracy: 0.64
```

```
Training model with max_features=log2
Accuracy: 0.64
```

```
Training model with max_features=None
Accuracy: 0.63
```

Random Forest Model Improvement

```
# List of max_depth values to experiment with
max_depth_values = [500, 100, 10, 50]

for max_depth in max_depth_values:
    print(f"Training model with max_depth={max_depth}")

    # Initialize the Random Forest model with different max_depth
    model = RandomForestClassifier(
        n_estimators=100,
        max_features='sqrt',
        max_depth=max_depth,
        random_state=42
    )
```

```
Training model with max_depth=500
Accuracy: 0.64
Training model with max_depth=100
Accuracy: 0.64
Training model with max_depth=10
Accuracy: 0.64
Training model with max_depth=50
Accuracy: 0.63
```

Method 3: XGBoost

DEPENDENCIES

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score
```

XGBOOST KNOWN AS EXTREME GRADIENT BOOSTING: A MACHINE LEARNING LIBRARY THAT USES GRADIENT BOOSTED DECISION TREES, **A SUPERVISED LEARNING BOOSTING ALGORITHM THAT MAKES USE OF GRADIENT DESCENT**. IT IS KNOWN FOR ITS SPEED, EFFICIENCY AND ABILITY TO SCALE WELL WITH LARGE DATASETS

GRID SEARCH IS USED TO FIND THE BEST COMBINATION OF **HYPERPARAMETERS**.

- TRIES DIFFERENT VALUES FOR:
  - N_ESTIMATORS (**NUMBER OF TREES**)
  - LEARNING_RATE (**ADJUSTS STEP SIZE**)
  - MAX_DEPTH (**TREE DEPTH**)
  - SUBSAMPLE (**PERCENTAGE OF SAMPLES USED PER TREE**)
  - COLSAMPLE_BYTREE (**PERCENTAGE OF FEATURES USED PER SPLIT**)

THE BEST MODEL IS SELECTED BASED ON HIGHEST ACCURACY.

```python
# Hyperparameter tuning using Grid Search
from sklearn.model_selection import GridSearchCV
```

```python
param_grid = {
    'n_estimators': [50, 100, 150, 200, 300],
    'learning_rate': [0.05, 0.1, .2],
    'max_depth': [1, 2, 4, 5, 6],
    'subsample': [.2, .5, 0.8, 1.0],
    'colsample_bytree': [.3, .6, 0.8, 1.0]
}
```

Results:

```
$ python exectest.py
Model Accuracy: 61.61%
Optimized Model Accuracy: 64.18%
Best Parameters: {'colsample_bytree': 0.6, 'learning_rate': 0.1, 'max_depth': 1, 'n_estimators': 200, 'subsample': 0.8}
```

Thank you for your time!