Assignment-3_Part-1

July 14, 2024

[17]:
```
#Assignment 3: Final Project (PART-1)
#Name: Uday Bhaskar Valapadasu
#ID: 11696364
```

[18]:
```
# Installing XBoost and SHAP Package in our Jupyter Notebook.
!pip install xgboost shap
```

```
Requirement already satisfied: xgboost in /opt/anaconda3/lib/python3.11/site-
packages (2.1.0)
Requirement already satisfied: shap in /opt/anaconda3/lib/python3.11/site-
packages (0.46.0)
Requirement already satisfied: numpy in /opt/anaconda3/lib/python3.11/site-
packages (from xgboost) (1.26.4)
Requirement already satisfied: scipy in /opt/anaconda3/lib/python3.11/site-
packages (from xgboost) (1.11.4)
Requirement already satisfied: scikit-learn in
/opt/anaconda3/lib/python3.11/site-packages (from shap) (1.2.2)
Requirement already satisfied: pandas in /opt/anaconda3/lib/python3.11/site-
packages (from shap) (2.1.4)
Requirement already satisfied: tqdm>=4.27.0 in
/opt/anaconda3/lib/python3.11/site-packages (from shap) (4.65.0)
Requirement already satisfied: packaging>20.9 in
/opt/anaconda3/lib/python3.11/site-packages (from shap) (23.1)
Requirement already satisfied: slicer==0.0.8 in
/opt/anaconda3/lib/python3.11/site-packages (from shap) (0.0.8)
Requirement already satisfied: numba in /opt/anaconda3/lib/python3.11/site-
packages (from shap) (0.59.0)
Requirement already satisfied: cloudpickle in
/opt/anaconda3/lib/python3.11/site-packages (from shap) (2.2.1)
Requirement already satisfied: llvmlite<0.43,>=0.42.0dev0 in
/opt/anaconda3/lib/python3.11/site-packages (from numba->shap) (0.42.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
/opt/anaconda3/lib/python3.11/site-packages (from pandas->shap) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/opt/anaconda3/lib/python3.11/site-packages (from pandas->shap) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in
/opt/anaconda3/lib/python3.11/site-packages (from pandas->shap) (2023.3)
Requirement already satisfied: joblib>=1.1.1 in
```

```
/opt/anaconda3/lib/python3.11/site-packages (from scikit-learn->shap) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/opt/anaconda3/lib/python3.11/site-packages (from scikit-learn->shap) (2.2.0)
Requirement already satisfied: six>=1.5 in /opt/anaconda3/lib/python3.11/site-
packages (from python-dateutil>=2.8.2->pandas->shap) (1.16.0)
```

[19]:
```
# Importing the necessary packages
import xgboost as xgb
import shap
from sklearn.metrics import accuracy_score, roc_curve, auc
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
from scipy import stats
import warnings
# Suppress the FutureWarning
warnings.simplefilter(action='ignore', category=FutureWarning)
```

[20]:
```
# Repeating the first 2 steps from assignment - 2
# Step 1: Imported the diabetes_df.csv from assignment-1
# Step 2: Using Pandas creating dataaframme frrom diabetes_df.csv and call it␣
 ↪assignment2_df

assignment2_df = pd.read_csv("diabetes_df.csv")
assignment2_df
```

[20]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
0	6	148	72	35	150	33.6	
1	1	85	66	29	150	26.6	
2	8	183	64	0	150	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	
..	…	…	…	…	…	…	
763	10	101	76	48	180	32.9	
764	2	122	70	27	150	36.8	
765	5	121	72	23	112	26.2	
766	1	126	60	0	150	30.1	
767	1	93	70	31	150	30.4	

	DiabetesPedigreeFunction	Age	Target
0	0.627	50	1
1	0.351	31	0
2	0.672	32	1
3	0.167	21	0
4	2.288	33	1
..	…	…	…
763	0.171	63	0

```
764                    0.340   27        0
765                    0.245   30        0
766                    0.349   47        1
767                    0.315   23        0

[768 rows x 9 columns]
```

[21]:
```
# Picked the split-1 (80-20) data split from the results of step-4 from␣
 ↪assignment-2

#features array
X = assignment2_df.drop(['Target'], axis=1)
#target array
y = assignment2_df['Target']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

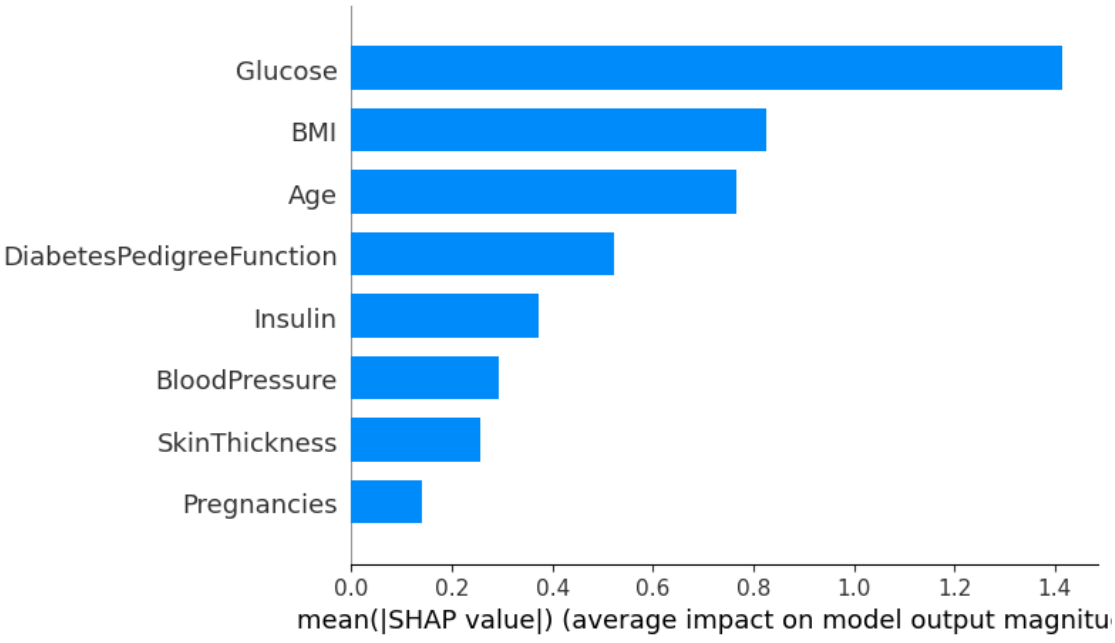
[22]:
```
# Lets creeate a default model using the command "model = xgb.XGClassifier()".
#You use your preferred for variable name in place of "model".

# Default XGBoost classifier model
fp_xgb = xgb.XGBClassifier()

# Fit the model to the training data
fp_xgb.fit(X_train, y_train)

# Make predictions on the test data
y_pred = fp_xgb.predict(X_test)
y_pred_proba_default = fp_xgb.predict_proba(X_test)[:, 1]
```

[23]:
```
# Optimizing the XGboost model using hyperparametrization
fp_xgb_opt = xgb.XGBClassifier(learning_rate=0.02, n_estimators=600,␣
 ↪objective='binary:logistic', nthread=1, random_state=42)
fp_xgb_opt.fit(X_train, y_train)

# Predict using optimized model
y_pred_opt = fp_xgb_opt.predict(X_test)
y_pred_proba_opt = fp_xgb_opt.predict_proba(X_test)[:, 1]
```

[24]:
```
# Calculate accuracies without and with optimization.

#This signifies accuracy without optimizing
accuracy_default = accuracy_score(y_test, y_pred_default)
# This signifies accuracy with optimizition
accuracy_opt = accuracy_score(y_test, y_pred_opt)
```

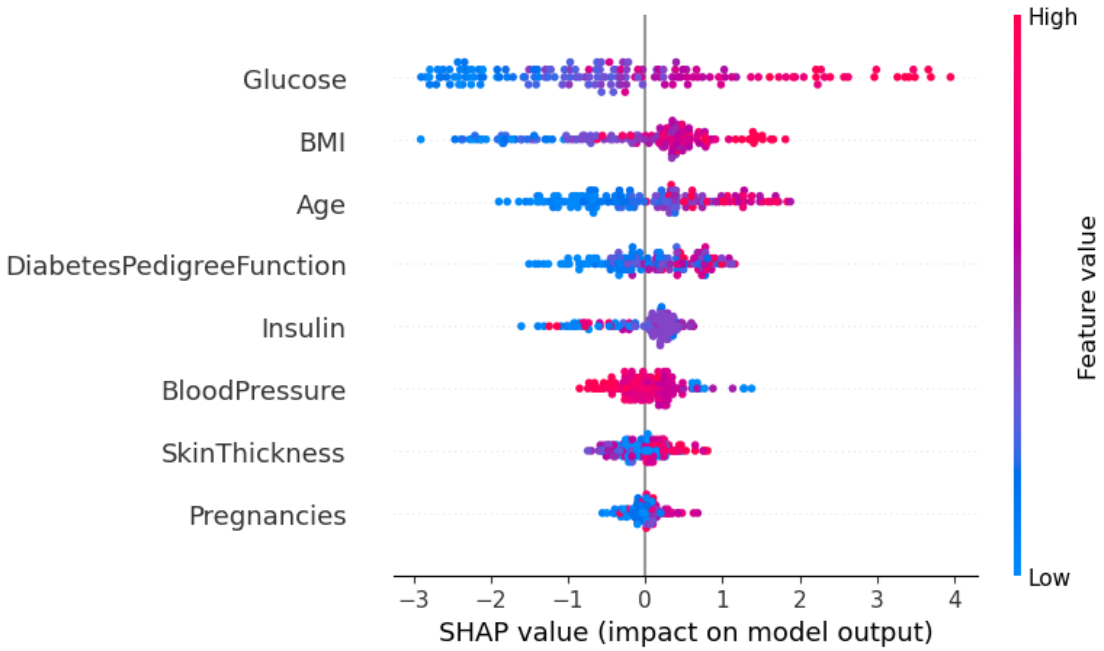
```
# Generate ROC curves for default and optimization.
fpr_default, tpr_default, _ = roc_curve(y_test, y_pred_proba_default)
fpr_opt, tpr_opt, _ = roc_curve(y_test, y_pred_proba_opt)
roc_auc_default = auc(fpr_default, tpr_default)
roc_auc_opt = auc(fpr_opt, tpr_opt)

# Calculate p-value for model acceptance
n = len(y_test)
k = sum(y_pred_opt == y_test)
p_value = stats.binomtest(k, n, p=0.5, alternative='greater').pvalue
```

[25]:
```
# Generating SHAP plots for the visualizing the model.
explainer = shap.TreeExplainer(fp_xgb_opt)
shap_values = explainer.shap_values(X_test)

# SHAP summary plot using Bar Plot
plt.figure(figsize=(10, 6))
shap.summary_plot(shap_values, X_test, plot_type="bar")
plt.title("SHAP Feature Importance")
plt.tight_layout()
plt.savefig("shap_feature_importance.png")
plt.close()
```

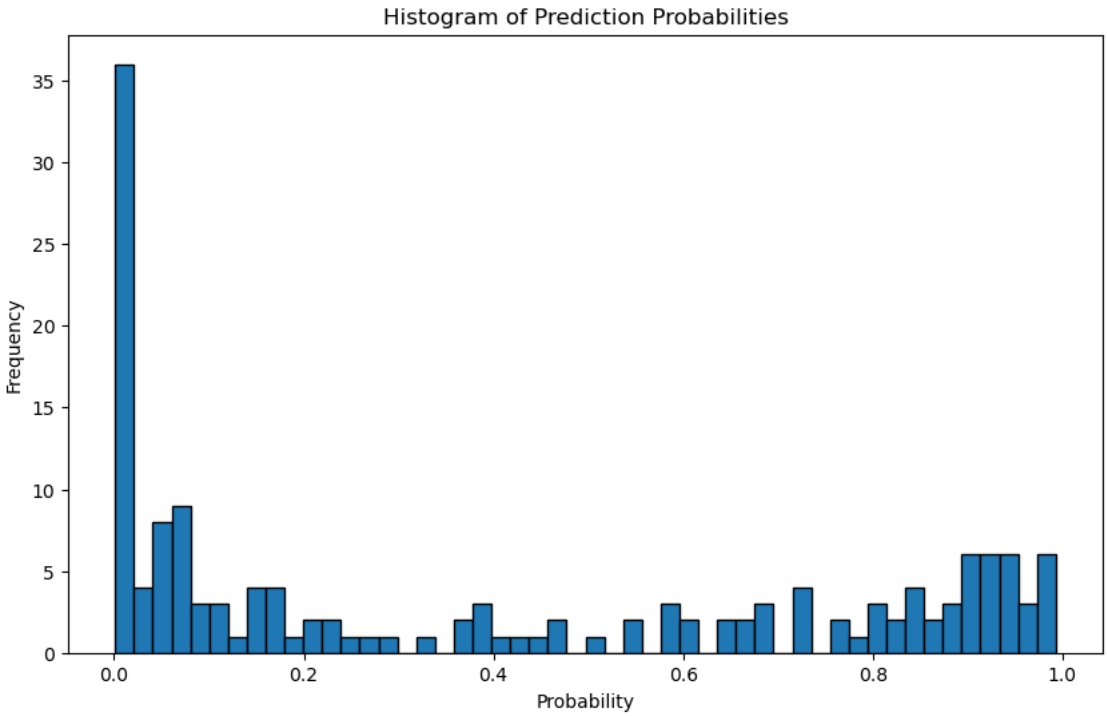


[26]:
```
# SHAP summary plot using Dot Plot
plt.figure(figsize=(10, 6))
shap.summary_plot(shap_values, X_test)
```

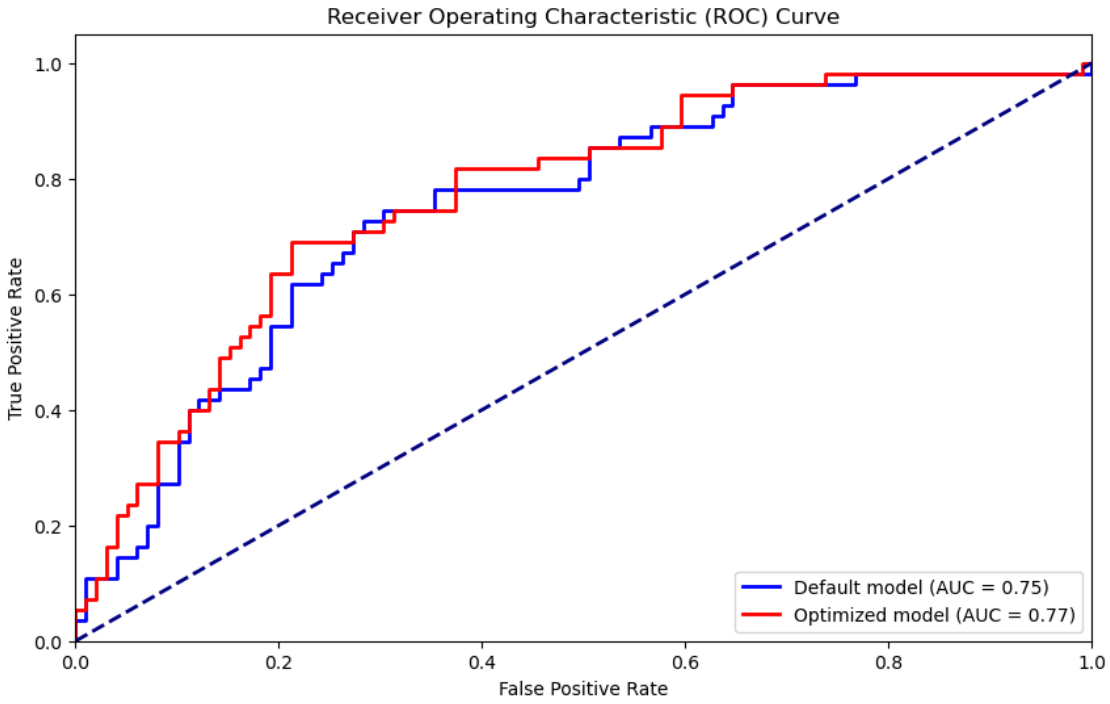
```
plt.title("SHAP Summary Plot")
plt.tight_layout()
plt.savefig("shap_summary_plot.png")
plt.close()
```



[27]:
```
# Generate histogram of prediction probabilities
plt.figure(figsize=(10, 6))
plt.hist(y_pred_proba_opt, bins=50, edgecolor='black')
plt.title("Histogram of Prediction Probabilities")
plt.xlabel("Probability")
plt.ylabel("Frequency")
plt.show()
plt.savefig("prediction_probabilities_histogram.png")
plt.close()
```



[28]:
```
# Generate ROC curve plot
plt.figure(figsize=(10, 6))
plt.plot(fpr_default, tpr_default, color='blue', lw=2, label=f'Default model␣
 ↪(AUC = {roc_auc_default:.2f})')
plt.plot(fpr_opt, tpr_opt, color='red', lw=2, label=f'Optimized model (AUC =␣
 ↪{roc_auc_opt:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
plt.savefig("roc_curve.png")
plt.close()
```



[29]:
```
# Print results
print(f"Default Model Accuracy: {accuracy_default:.4f}")
print(f"Optimized Model Accuracy: {accuracy_opt:.4f}")
print(f"Default Model AUC: {roc_auc_default:.4f}")
print(f"Optimized Model AUC: {roc_auc_opt:.4f}")
print(f"P-value: {p_value:.4f}")
```

```
Default Model Accuracy: 0.7078
Optimized Model Accuracy: 0.7273
Default Model AUC: 0.7519
Optimized Model AUC: 0.7728
P-value: 0.0000
```