parkinsons-disease-analysis

June 26, 2024

[2]:
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import ipywidgets as widgets
import seaborn as sns


from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from xgboost import XGBClassifier
from sklearn.metrics import classification_report, roc_auc_score,␣
 ↪confusion_matrix, precision_recall_curve, average_precision_score
from sklearn.model_selection import learning_curve
```

[3]:
```
df = pd.read_csv('parkinsons_disease_data.csv')
```

[4]:
```
df.head()
```

[4]:

	PatientID	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	\
0	3058	85	0	3	1	19.619878	0	
1	3059	75	0	0	2	16.247339	1	
2	3060	70	1	0	0	15.368239	0	
3	3061	52	0	0	0	15.454557	0	
4	3062	87	0	0	1	18.616042	0	

	AlcoholConsumption	PhysicalActivity	DietQuality	…	\
0	5.108241	1.380660	3.893969	…	
1	6.027648	8.409804	8.513428	…	
2	2.242135	0.213275	6.498805	…	
3	5.997788	1.375045	6.715033	…	
4	9.775243	1.188607	4.657572	…	

	FunctionalAssessment	Tremor	Rigidity	Bradykinesia	PosturalInstability	\
0	1.572427	1	0	0	0	
1	4.787551	0	1	0	1	
2	2.130686	1	0	0	0	

```
3              3.391288       1         1             1                    0
4              3.200969       0         0             0                    1

   SpeechProblems  SleepDisorders  Constipation  Diagnosis  DoctorInCharge
0               0               0             0          0     DrXXXConfid
1               0               1             0          1     DrXXXConfid
2               1               0             1          1     DrXXXConfid
3               0               0             1          1     DrXXXConfid
4               0               1             0          0     DrXXXConfid

[5 rows x 35 columns]
```

[5]:
```
def get_df_info(df):
    print("\n\033[1mShape of DataFrame:\033[0m ", df.shape)
    print("\n\033[1mColumns in DataFrame:\033[0m ", df.columns.to_list())
    print("\n\033[1mData types of columns:\033[0m\n", df.dtypes)

    print("\n\033[1mInformation about DataFrame:\033[0m")
    df.info()

    print("\n\033[1mNumber of unique values in each column:\033[0m")
    for col in df.columns:
        print(f"\033[1m{col}\033[0m: {df[col].nunique()}")

    print("\n\033[1mNumber of null values in each column:\033[0m\n", df.
 ↪isnull().sum())

    print("\n\033[1mNumber of duplicate rows:\033[0m ", df.duplicated().sum())

    print("\n\033[1mDescriptive statistics of DataFrame:\033[0m\n", df.
 ↪describe().transpose())

# Call the function
get_df_info(df)
```

Shape of DataFrame: (2105, 35)

Columns in DataFrame: ['PatientID', 'Age', 'Gender', 'Ethnicity', 'EducationLevel', 'BMI', 'Smoking', 'AlcoholConsumption', 'PhysicalActivity', 'DietQuality', 'SleepQuality', 'FamilyHistoryParkinsons', 'TraumaticBrainInjury', 'Hypertension', 'Diabetes', 'Depression', 'Stroke', 'SystolicBP', 'DiastolicBP', 'CholesterolTotal', 'CholesterolLDL', 'CholesterolHDL', 'CholesterolTriglycerides', 'UPDRS', 'MoCA', 'FunctionalAssessment', 'Tremor', 'Rigidity', 'Bradykinesia', 'PosturalInstability', 'SpeechProblems', 'SleepDisorders', 'Constipation', 'Diagnosis', 'DoctorInCharge']

Data types of columns:

```
 PatientID                     int64
Age                           int64
Gender                        int64
Ethnicity                     int64
EducationLevel                int64
BMI                         float64
Smoking                       int64
AlcoholConsumption          float64
PhysicalActivity            float64
DietQuality                 float64
SleepQuality                float64
FamilyHistoryParkinsons       int64
TraumaticBrainInjury          int64
Hypertension                  int64
Diabetes                      int64
Depression                    int64
Stroke                        int64
SystolicBP                    int64
DiastolicBP                   int64
CholesterolTotal            float64
CholesterolLDL              float64
CholesterolHDL              float64
CholesterolTriglycerides    float64
UPDRS                       float64
MoCA                        float64
FunctionalAssessment        float64
Tremor                        int64
Rigidity                      int64
Bradykinesia                  int64
PosturalInstability           int64
SpeechProblems                int64
SleepDisorders                int64
Constipation                  int64
Diagnosis                     int64
DoctorInCharge               object
dtype: object
```

Information about DataFrame:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2105 entries, 0 to 2104
Data columns (total 35 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   PatientID                  2105 non-null   int64
 1   Age                        2105 non-null   int64
 2   Gender                     2105 non-null   int64
```

```
 3   Ethnicity                2105 non-null   int64
 4   EducationLevel           2105 non-null   int64
 5   BMI                      2105 non-null   float64
 6   Smoking                  2105 non-null   int64
 7   AlcoholConsumption       2105 non-null   float64
 8   PhysicalActivity         2105 non-null   float64
 9   DietQuality              2105 non-null   float64
 10  SleepQuality             2105 non-null   float64
 11  FamilyHistoryParkinsons  2105 non-null   int64
 12  TraumaticBrainInjury     2105 non-null   int64
 13  Hypertension             2105 non-null   int64
 14  Diabetes                 2105 non-null   int64
 15  Depression               2105 non-null   int64
 16  Stroke                   2105 non-null   int64
 17  SystolicBP               2105 non-null   int64
 18  DiastolicBP              2105 non-null   int64
 19  CholesterolTotal         2105 non-null   float64
 20  CholesterolLDL           2105 non-null   float64
 21  CholesterolHDL           2105 non-null   float64
 22  CholesterolTriglycerides 2105 non-null   float64
 23  UPDRS                    2105 non-null   float64
 24  MoCA                     2105 non-null   float64
 25  FunctionalAssessment     2105 non-null   float64
 26  Tremor                   2105 non-null   int64
 27  Rigidity                 2105 non-null   int64
 28  Bradykinesia             2105 non-null   int64
 29  PosturalInstability      2105 non-null   int64
 30  SpeechProblems           2105 non-null   int64
 31  SleepDisorders           2105 non-null   int64
 32  Constipation             2105 non-null   int64
 33  Diagnosis                2105 non-null   int64
 34  DoctorInCharge           2105 non-null   object
dtypes: float64(12), int64(22), object(1)
memory usage: 575.7+ KB
```

Number of unique values in each column:

PatientID: 2105
Age: 40
Gender: 2
Ethnicity: 4
EducationLevel: 4
BMI: 2105
Smoking: 2
AlcoholConsumption: 2105
PhysicalActivity: 2105
DietQuality: 2105
SleepQuality: 2105
FamilyHistoryParkinsons: 2

TraumaticBrainInjury: 2
Hypertension: 2
Diabetes: 2
Depression: 2
Stroke: 2
SystolicBP: 90
DiastolicBP: 60
CholesterolTotal: 2105
CholesterolLDL: 2105
CholesterolHDL: 2105
CholesterolTriglycerides: 2105
UPDRS: 2105
MoCA: 2105
FunctionalAssessment: 2105
Tremor: 2
Rigidity: 2
Bradykinesia: 2
PosturalInstability: 2
SpeechProblems: 2
SleepDisorders: 2
Constipation: 2
Diagnosis: 2
DoctorInCharge: 1

Number of null values in each column:

PatientID	0
Age	0
Gender	0
Ethnicity	0
EducationLevel	0
BMI	0
Smoking	0
AlcoholConsumption	0
PhysicalActivity	0
DietQuality	0
SleepQuality	0
FamilyHistoryParkinsons	0
TraumaticBrainInjury	0
Hypertension	0
Diabetes	0
Depression	0
Stroke	0
SystolicBP	0
DiastolicBP	0
CholesterolTotal	0
CholesterolLDL	0
CholesterolHDL	0
CholesterolTriglycerides	0

```
UPDRS                        0
MoCA                         0
FunctionalAssessment         0
Tremor                       0
Rigidity                     0
Bradykinesia                 0
PosturalInstability          0
SpeechProblems               0
SleepDisorders               0
Constipation                 0
Diagnosis                    0
DoctorInCharge               0
dtype: int64
```

Number of duplicate rows: 0

Descriptive statistics of DataFrame:

	count	mean	std	min	\
PatientID	2105.0	4110.000000	607.805479	3058.000000	
Age	2105.0	69.601900	11.594511	50.000000	
Gender	2105.0	0.492637	0.500065	0.000000	
Ethnicity	2105.0	0.692637	1.003827	0.000000	
EducationLevel	2105.0	1.337292	0.895840	0.000000	
BMI	2105.0	27.209493	7.208099	15.008333	
Smoking	2105.0	0.296437	0.456795	0.000000	
AlcoholConsumption	2105.0	10.040413	5.687014	0.002228	
PhysicalActivity	2105.0	5.016674	2.890919	0.004157	
DietQuality	2105.0	4.912901	2.872115	0.000011	
SleepQuality	2105.0	6.996639	1.753065	4.000497	
FamilyHistoryParkinsons	2105.0	0.145843	0.353033	0.000000	
TraumaticBrainInjury	2105.0	0.106413	0.308439	0.000000	
Hypertension	2105.0	0.145843	0.353033	0.000000	
Diabetes	2105.0	0.148219	0.355401	0.000000	
Depression	2105.0	0.205226	0.403962	0.000000	
Stroke	2105.0	0.048931	0.215775	0.000000	
SystolicBP	2105.0	133.719715	26.502355	90.000000	
DiastolicBP	2105.0	90.249881	17.061488	60.000000	
CholesterolTotal	2105.0	226.860840	43.589406	150.062698	
CholesterolLDL	2105.0	126.147858	43.407036	50.022828	
CholesterolHDL	2105.0	59.670352	23.370920	20.027981	
CholesterolTriglycerides	2105.0	222.940500	101.895822	50.113604	
UPDRS	2105.0	101.415318	56.591448	0.028441	
MoCA	2105.0	15.094314	8.643014	0.021191	
FunctionalAssessment	2105.0	4.989694	2.933877	0.001505	
Tremor	2105.0	0.431829	0.495449	0.000000	
Rigidity	2105.0	0.252732	0.434682	0.000000	
Bradykinesia	2105.0	0.207601	0.405686	0.000000	
PosturalInstability	2105.0	0.138717	0.345733	0.000000	

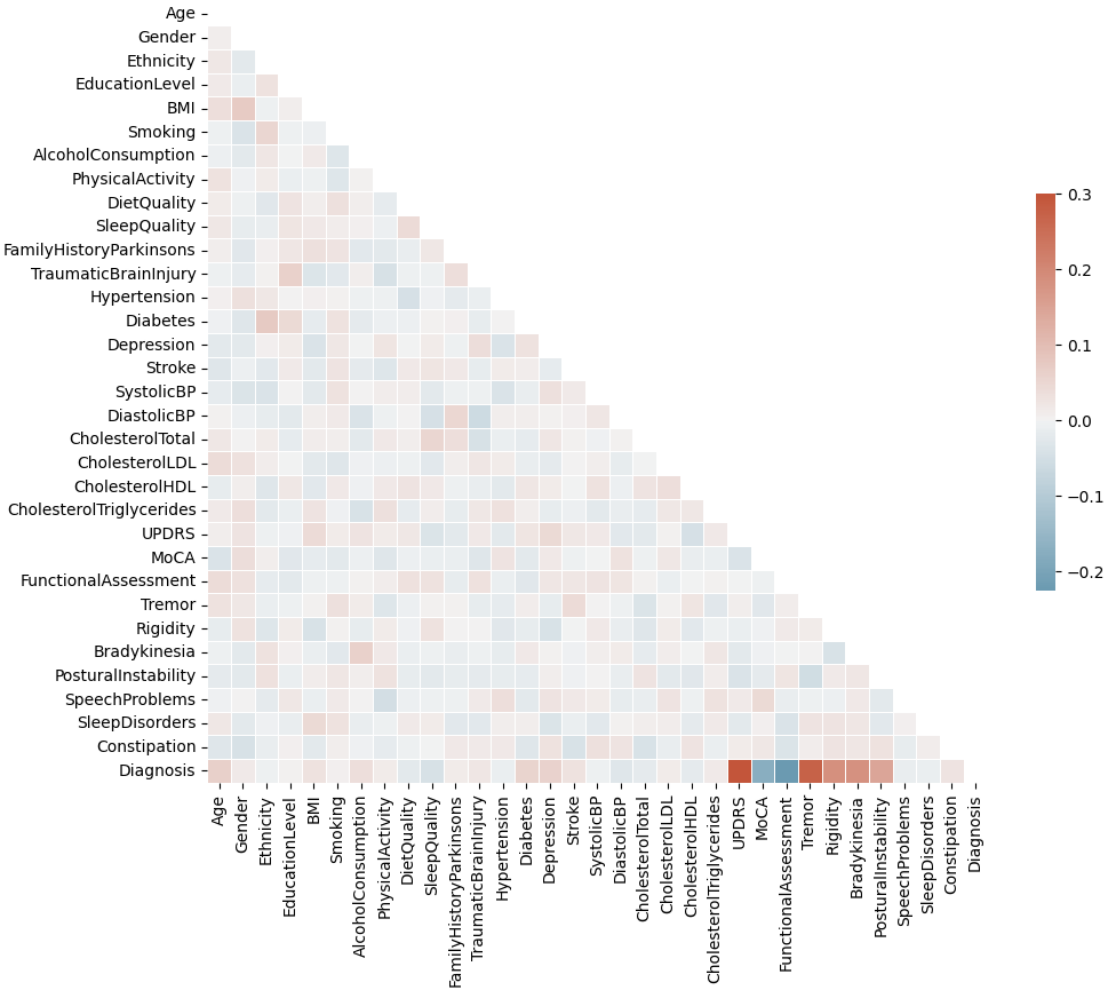
SpeechProblems	2105.0	0.295012	0.456156	0.000000
SleepDisorders	2105.0	0.245131	0.430267	0.000000
Constipation	2105.0	0.296912	0.457006	0.000000
Diagnosis	2105.0	0.619477	0.485631	0.000000

	25%	50%	75%	max
PatientID	3584.000000	4110.000000	4636.000000	5162.000000
Age	60.000000	70.000000	80.000000	89.000000
Gender	0.000000	0.000000	1.000000	1.000000
Ethnicity	0.000000	0.000000	1.000000	3.000000
EducationLevel	1.000000	1.000000	2.000000	3.000000
BMI	20.782176	27.184571	33.462452	39.999887
Smoking	0.000000	0.000000	1.000000	1.000000
AlcoholConsumption	5.150278	10.070337	14.829565	19.988866
PhysicalActivity	2.455703	5.031550	7.512795	9.995255
DietQuality	2.478503	4.825187	7.381487	9.995864
SleepQuality	5.488864	6.929819	8.558719	9.999821
FamilyHistoryParkinsons	0.000000	0.000000	0.000000	1.000000
TraumaticBrainInjury	0.000000	0.000000	0.000000	1.000000
Hypertension	0.000000	0.000000	0.000000	1.000000
Diabetes	0.000000	0.000000	0.000000	1.000000
Depression	0.000000	0.000000	0.000000	1.000000
Stroke	0.000000	0.000000	0.000000	1.000000
SystolicBP	110.000000	133.000000	157.000000	179.000000
DiastolicBP	75.000000	91.000000	105.000000	119.000000
CholesterolTotal	189.385178	228.528256	264.608100	299.963074
CholesterolLDL	88.841960	126.884570	163.912782	199.985981
CholesterolHDL	39.538643	59.343357	79.366628	99.982265
CholesterolTriglycerides	132.520174	222.802452	311.699109	399.975022
UPDRS	53.048148	102.561023	149.831682	198.953604
MoCA	7.517160	14.963574	22.608362	29.970107
FunctionalAssessment	2.415890	4.983227	7.484220	9.992697
Tremor	0.000000	0.000000	1.000000	1.000000
Rigidity	0.000000	0.000000	1.000000	1.000000
Bradykinesia	0.000000	0.000000	0.000000	1.000000
PosturalInstability	0.000000	0.000000	0.000000	1.000000
SpeechProblems	0.000000	0.000000	1.000000	1.000000
SleepDisorders	0.000000	0.000000	0.000000	1.000000
Constipation	0.000000	0.000000	1.000000	1.000000
Diagnosis	0.000000	1.000000	1.000000	1.000000

[9]:
```
df = df.drop(['DoctorInCharge', 'PatientID'], axis=1)
```

[10]:
```
# Compute the correlation matrix
corr = df.corr(numeric_only=True)

# Generate a mask for the upper triangle
```

```
mask = np.triu(np.ones_like(corr, dtype=bool))

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(230, 20, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})

plt.show()
```



[11]:
```
# Separating Features(X) and Target(y) variables
X = df.drop(['Diagnosis'], axis =1)
```

```
y = df['Diagnosis']
```

[12]:
```
# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=101)

# Feature Scaling
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[13]:
```
# XGBoost Classifier
xgb_classifier = XGBClassifier(
    objective='binary:logistic',  # Binary classification objective
    max_depth=3,                  # Maximum depth of trees
    learning_rate=0.1,            # Learning rate (step size)
    n_estimators=100,             # Number of boosting round
)
```

[14]:
```
xgb_classifier.fit(X_train_scaled, y_train)
```

[14]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.1, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=3, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=100, n_jobs=None,
              num_parallel_tree=None, random_state=None, …)
```

[15]:
```
y_pred = xgb_classifier.predict(X_test_scaled)
```

[16]:
```
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.90      0.92       173
           1       0.93      0.96      0.95       248

    accuracy                           0.94       421
   macro avg       0.94      0.93      0.93       421
weighted avg       0.94      0.94      0.94       421
```

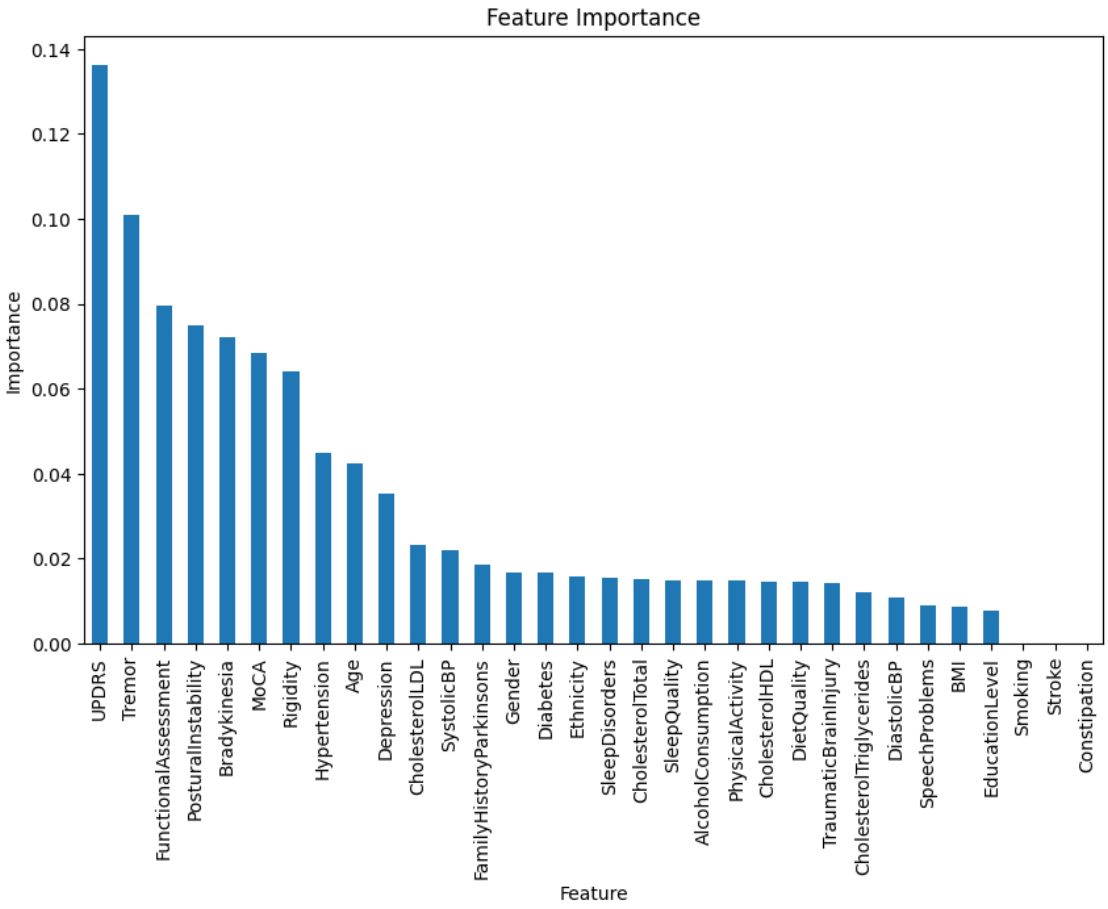
[17]:
```
roc_auc = roc_auc_score(y_test, y_pred)
print(f"AUC-ROC Score: {roc_auc:.4f}")
```

```
AUC-ROC Score: 0.9298
```

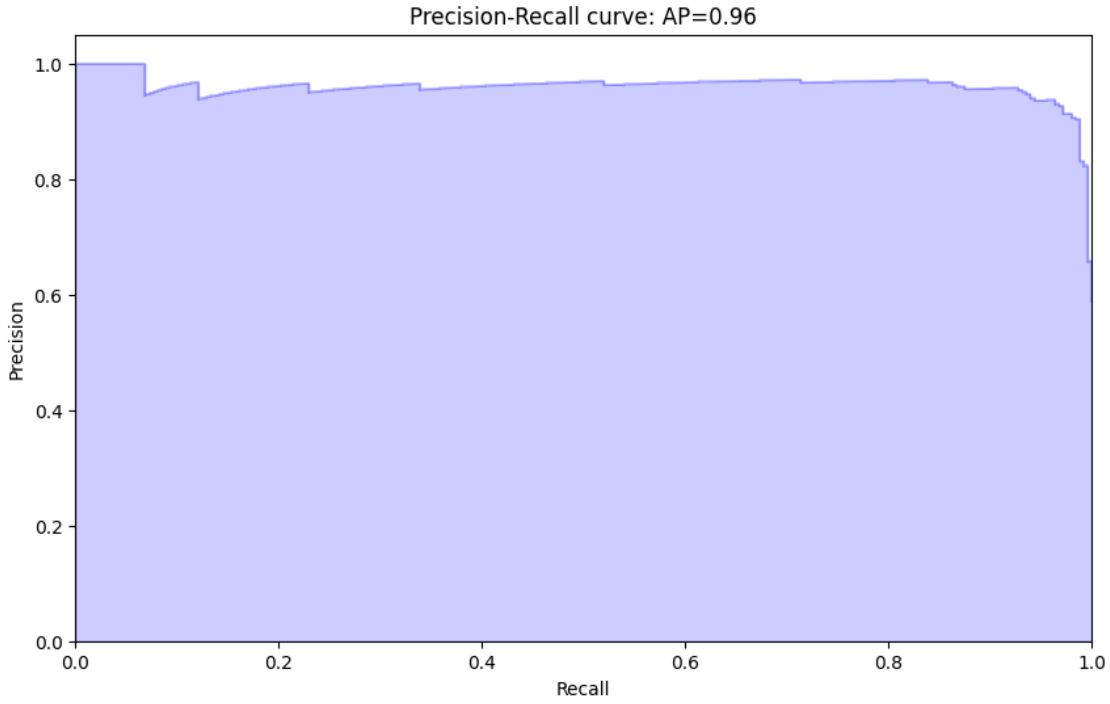
[18]:
```
# Confusion Matrix

conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
```

```
Confusion Matrix:
[[155  18]
 [  9 239]]
```

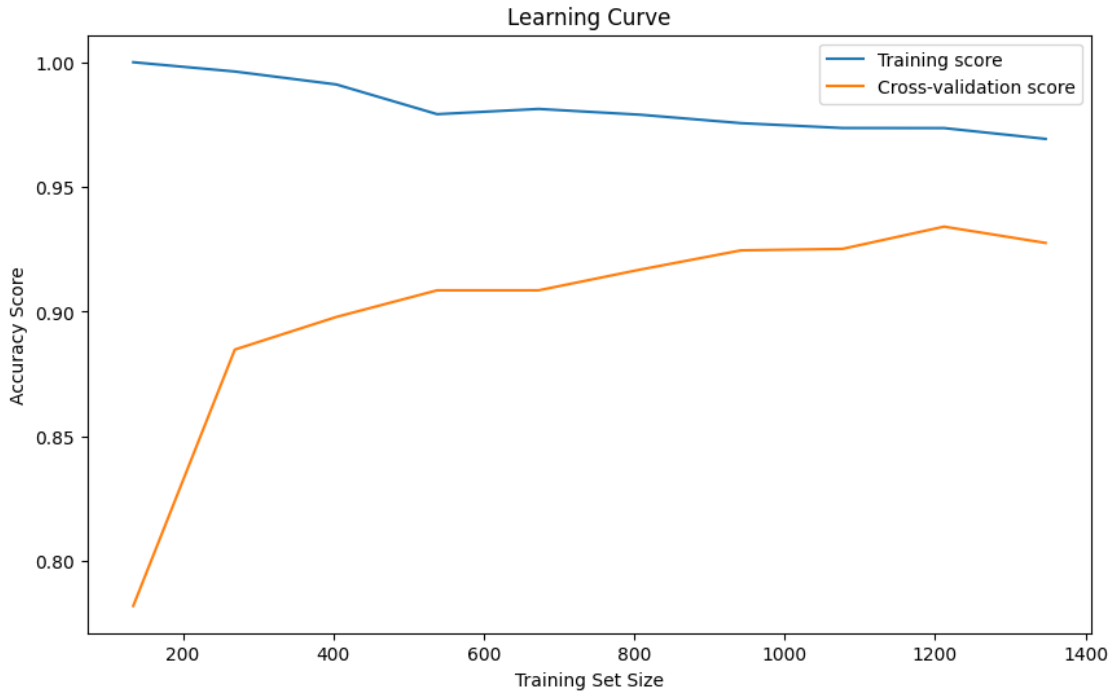
[19]:
```
# Feature Importance

feature_importance = pd.Series(xgb_classifier.feature_importances_, index=X.
  ↪columns)
feature_importance.sort_values(ascending=False, inplace=True)

# Plot feature importances

plt.figure(figsize=(10, 6))
feature_importance.plot(kind='bar')
plt.xlabel('Feature')
plt.ylabel('Importance')
plt.title('Feature Importance')
plt.show()
```



[20]:
```
# Precision_Recall_Curve


y_pred_proba = xgb_classifier.predict_proba(X_test_scaled)[:, 1]
precision, recall, _ = precision_recall_curve(y_test, y_pred_proba)
average_precision = average_precision_score(y_test, y_pred_proba)

plt.figure(figsize=(10, 6))
plt.step(recall, precision, color='b', alpha=0.2, where='post')
plt.fill_between(recall, precision, step='post', alpha=0.2, color='b')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.ylim([0.0, 1.05])
plt.xlim([0.0, 1.0])
plt.title(f'Precision-Recall curve: AP={average_precision:0.2f}')
plt.show()
```



[21]:
```
# Learning Curve

train_sizes, train_scores, valid_scores = learning_curve(
    xgb_classifier, X_train_scaled, y_train, train_sizes=np.linspace(0.1, 1.0,␣
 ↪10),
    cv=5, scoring='accuracy'
)

plt.figure(figsize=(10, 6))
plt.plot(train_sizes, np.mean(train_scores, axis=1), label='Training score')
plt.plot(train_sizes, np.mean(valid_scores, axis=1), label='Cross-validation␣
 ↪score')
plt.title('Learning Curve')
plt.xlabel('Training Set Size')
plt.ylabel('Accuracy Score')
plt.legend()
plt.show()
```



[22]:
```
# Distribution of Features

import matplotlib.pyplot as plt
import ipywidgets as widgets
import seaborn as sns

# Define a function to plot the disctribution of features
def plot_feature(feature):
    plt.figure(figsize=(10,6))
    df[feature].hist(bins=30)
    plt.title(f'Distribution of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Frequency')
    plt.show()

# Create a dropdown widget with the dataframe's column names
dropdown = widgets.Dropdown(options=df.columns, description='Feature:')

# Use the interact function to create the widget and the plot
widgets.interact(plot_feature, feature=dropdown);
```

```
interactive(children=(Dropdown(description='Feature:', options=('Age', 'Gender',␣
↪'Ethnicity', 'EducationLevel'…
```