```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from xgboost import XGBClassifier
from sklearn.model_selection import GridSearchCV
# Load the data
df = pd.read_csv('/content/new_model.csv')
df
```

```
df.isna().sum()
```

```
Bp        0
Sg        0
Al        0
Su        0
```

```
Rbc       0
Bu        0
Sc        0
Sod       0
Pot       0
Hemo      0
Wbcc      0
Rbcc      0
Htn       0
Class     0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 14 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Bp      400 non-null    float64
 1   Sg      400 non-null    float64
 2   Al      400 non-null    float64
 3   Su      400 non-null    float64
 4   Rbc     400 non-null    float64
 5   Bu      400 non-null    float64
 6   Sc      400 non-null    float64
 7   Sod     400 non-null    float64
 8   Pot     400 non-null    float64
 9   Hemo    400 non-null    float64
 10  Wbcc    400 non-null    float64
 11  Rbcc    400 non-null    float64
 12  Htn     400 non-null    float64
 13  Class   400 non-null    int64  
dtypes: float64(13), int64(1)
memory usage: 43.9 KB
```

```
df.shape
```

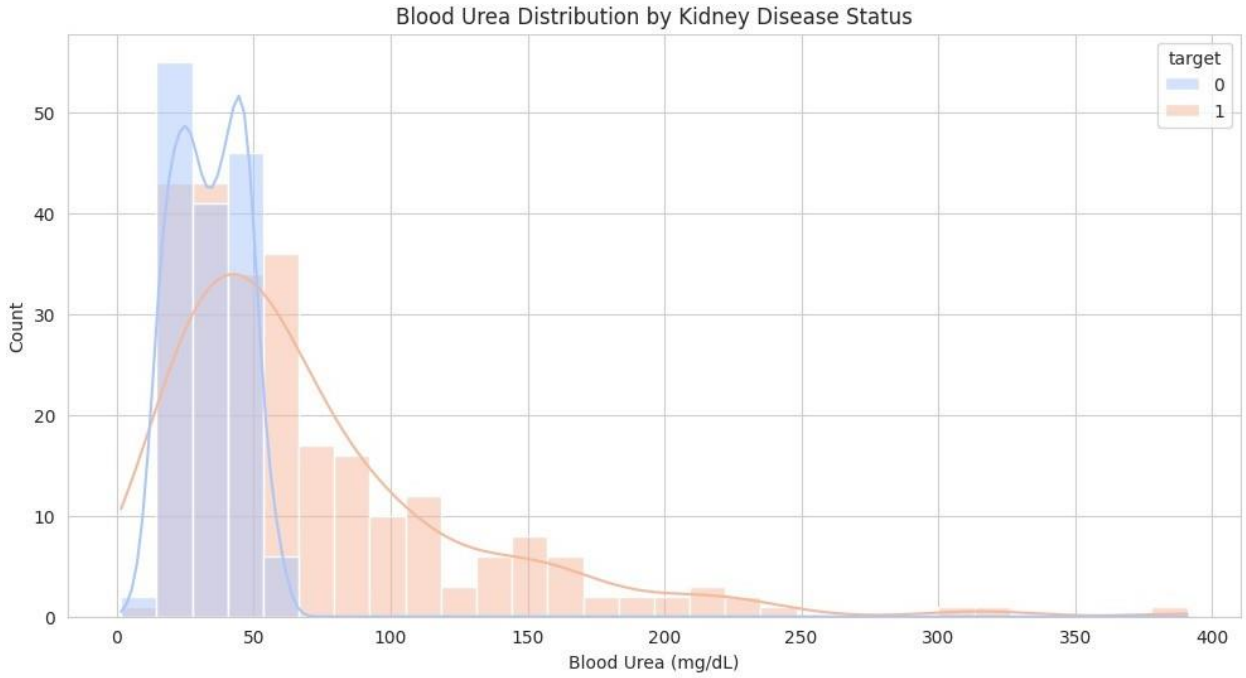
```
(400, 14)
```

```
df.describe()
```

```
df = df.rename(
    columns={
        'Bp': 'bp',
        'Sg': 'sp_g',
        'Al': 'alb',
        'Su': 'sugar',
        'Bu': 'blood_urea',
        'Sc': 'ser_creat',
        'Sod': 'sodium',
        'Pot': 'potassium',
        'Hemo': 'hb',
        'Htn': 'hp_tn',
        'Class': 'target'
```

```
    },
    errors="raise"
)

plt.figure(figsize=(12, 6))
sns.histplot(data=df, x='blood_urea', hue='target', bins=30, kde=True,
palette='coolwarm')
plt.title('Blood Urea Distribution by Kidney Disease Status')
plt.xlabel('Blood Urea (mg/dL)')
plt.ylabel('Count')
```

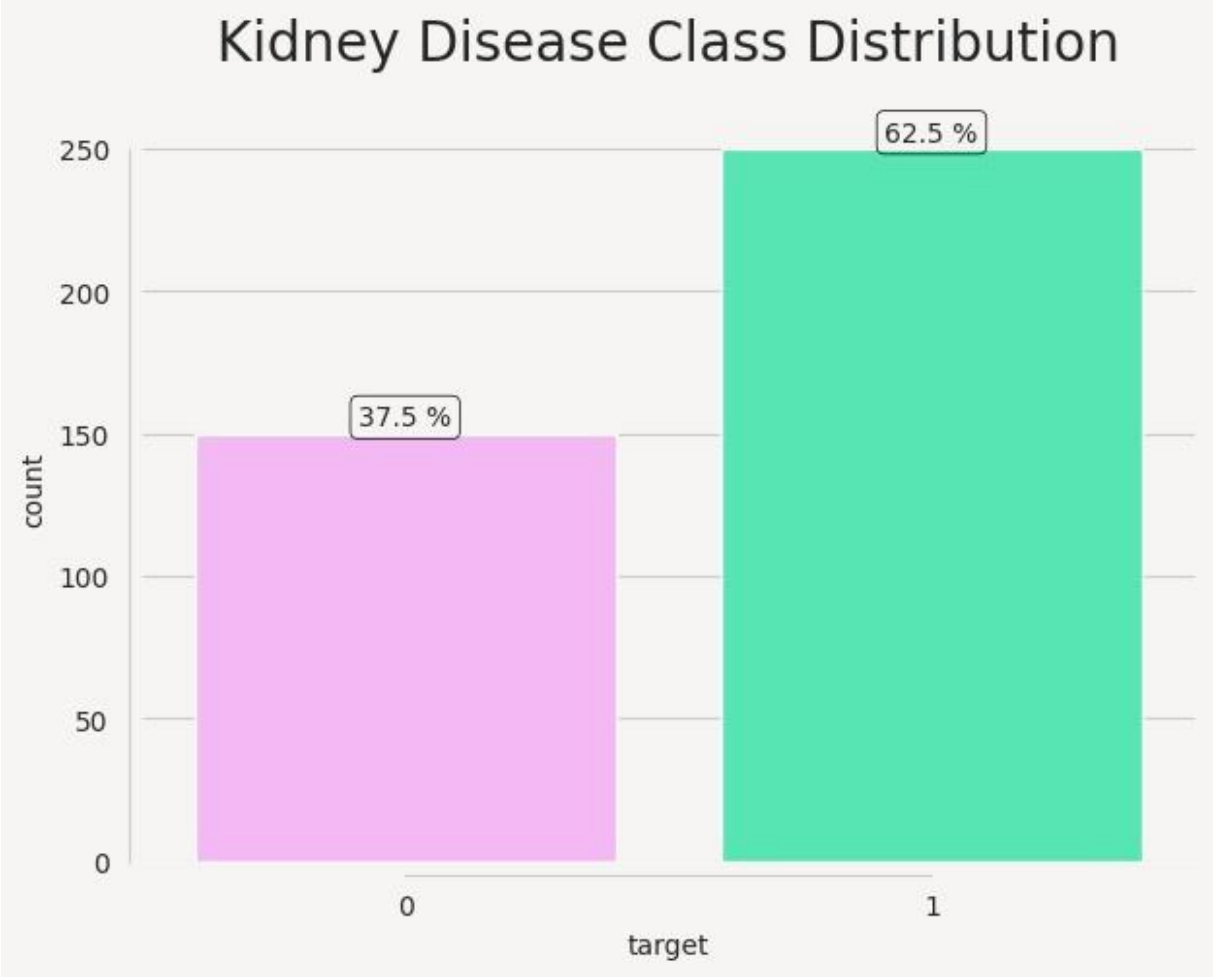
```
plt.grid(True)
plt.show()
```



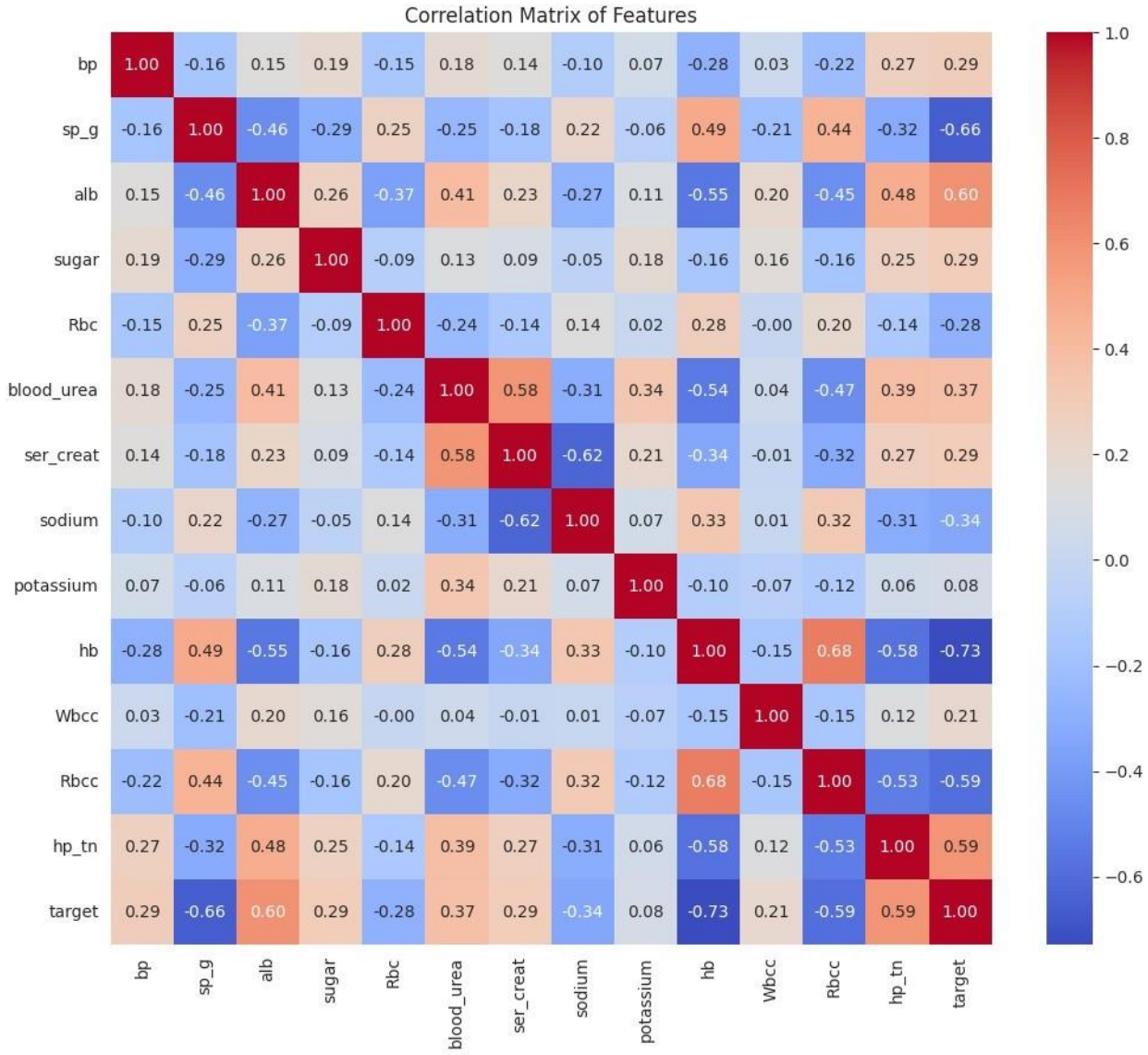

```
# Custom color palette
mypal = ['#FC05FB', '#FEAEFE', '#FCD2FC', '#F3FEFA', '#B4FFE4',
'#3FFEBA']

plt.figure(figsize=(7, 5), facecolor='#F6F5F4')  # Figure size and
background
total = float(len(df))  # Total number of patients

# Countplot for Class (0 = no kidney disease, 1 = disease)
ax = sns.countplot(x='target', data=df, palette=mypal[1::4])
ax.set_facecolor('#F6F5F4')  # Match axes background

# Add % labels on bars
for p in ax.patches:
    height = p.get_height()
    ax.text(
        p.get_x() + p.get_width() / 2.,
        height + 3,
        '{:1.1f} %'.format((height / total) * 100),
        ha="center",
        bbox=dict(facecolor='none', edgecolor='black',
boxstyle='round', linewidth=0.5)
    )

# Title and cleanup
```

```
ax.set_title('Kidney Disease Class Distribution', fontsize=20, y=1.05)
sns.despine(right=True)
sns.despine(offset=5, trim=True)
plt.show()
```

```
<ipython-input-88-127e8330ad7f>:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  ax = sns.countplot(x='target', data=df, palette=mypal[1::4])
```

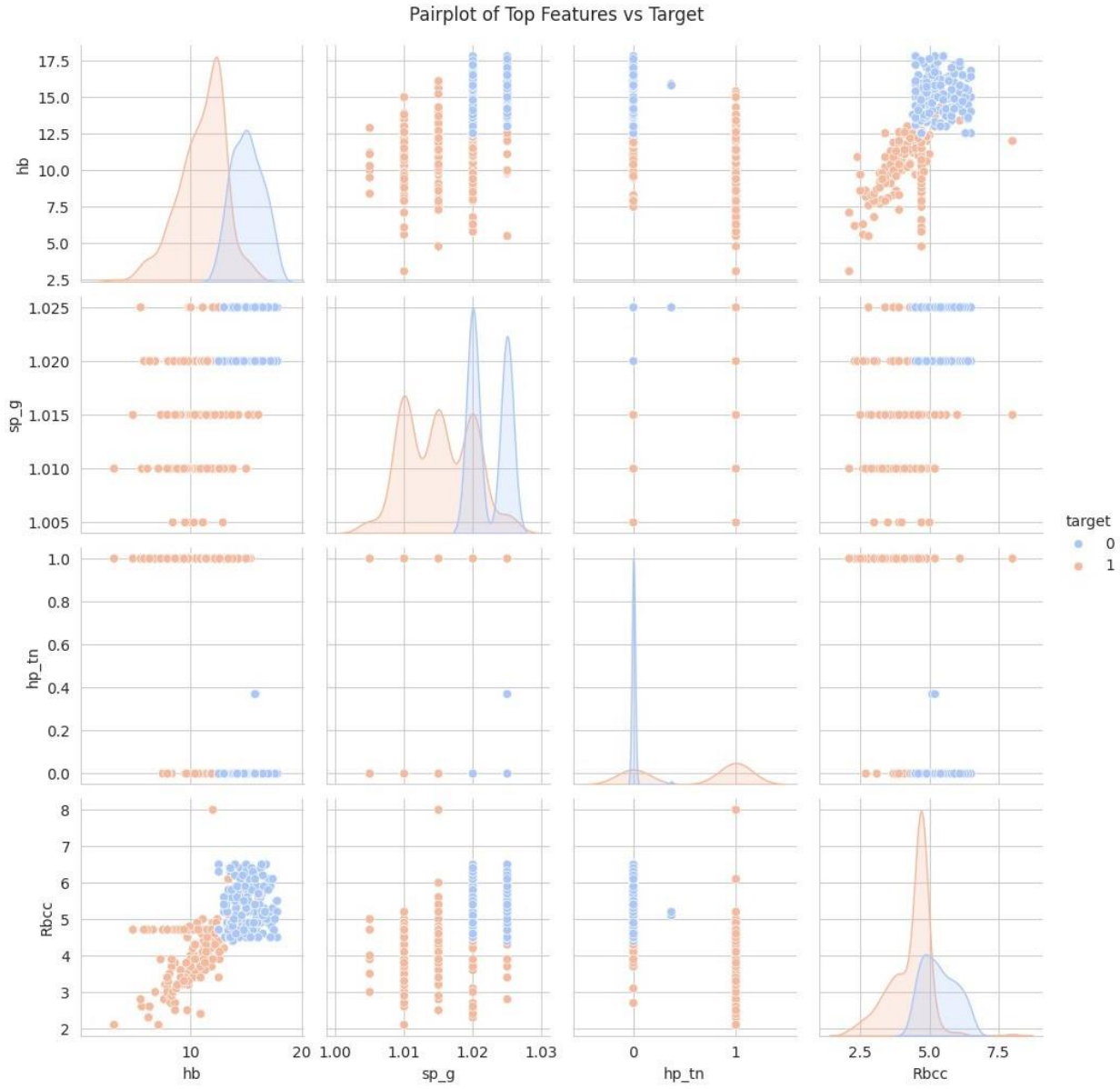



```
plt.figure(figsize=(12, 10))
corr = df.corr()
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix of Features')
plt.show()
```




```
# Separate features and target
X = df.drop('target', axis=1)
y = df['target']
# Feature Importance
feature_importance = pd.Series(model.feature_importances_,
index=X.columns).sort_values(ascending=False)

# Select Top 4 important features
top_features = feature_importance.head(4).index.tolist()

print("Selected Features for Pairplot:", top_features)

# Pairplot for Top Features
sns.pairplot(df[top_features + ['target']], hue='target',
```

```
diag_kind='kde', palette='coolwarm')
plt.suptitle('Pairplot of Top Features vs Target', y=1.02)
plt.show()

Selected Features for Pairplot: ['hb', 'sp_g', 'hp_tn', 'Rbcc']
```




```
import seaborn as sns
import matplotlib.pyplot as plt

# Updated list of actual numerical features
numerical_features = ['bp', 'blood_urea', 'ser_creat', 'sodium',
'potassium', 'hb']
```

```
# Create the pairplot
plt.figure(figsize=(20, 20))
sns.set_style("whitegrid")

plot = sns.pairplot(df[numerical_features + ['target']],
                    hue='target',
                    palette={0: 'red', 1: 'green'},
                    plot_kws={'alpha': 0.6, 's': 30},
                    diag_kind='kde',
                    corner=True)

# Set title
plot.fig.suptitle('Scatter Matrix of Kidney Disease Features (Red=No
Disease, Green=Disease)',
                  y=1.02, size=16)

# Custom legend
handles = plot._legend_data.values()
labels = ['No Kidney Disease', 'Kidney Disease']
plot._legend.remove()
plot.fig.legend(handles=handles, labels=labels,
                loc='upper right', bbox_to_anchor=(0.9, 0.9),
                title='Class')

plt.tight_layout()
plt.show()
```

```
<Figure size 2000x2000 with 0 Axes>
```


```
# Split into features and target
X = df.drop('target', axis=1)
y = df['target']

# Split into train and test sets with stratified sampling
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y)


# Standardize the features
scaler = StandardScaler()
```

```
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

logistic regression

```
lr = LogisticRegression(max_iter=1000)
lr.fit(X_train_scaled, y_train)
y_pred_lr = lr.predict(X_test_scaled)
print("Logistic Regression Accuracy:", accuracy_score(y_test,
y_pred_lr))
```

```
Logistic Regression Accuracy: 0.975
```

```
svm = SVC(probability=True)
svm.fit(X_train_scaled, y_train)
y_pred_svm = svm.predict(X_test_scaled)
print("SVM Accuracy:", accuracy_score(y_test, y_pred_svm))
```

```
SVM Accuracy: 0.975
```

```
xgb = XGBClassifier(use_label_encoder=False, eval_metric='logloss')
xgb.fit(X_train_scaled, y_train)
y_pred_xgb = xgb.predict(X_test_scaled)
print("XGBoost Accuracy:", accuracy_score(y_test, y_pred_xgb))
```

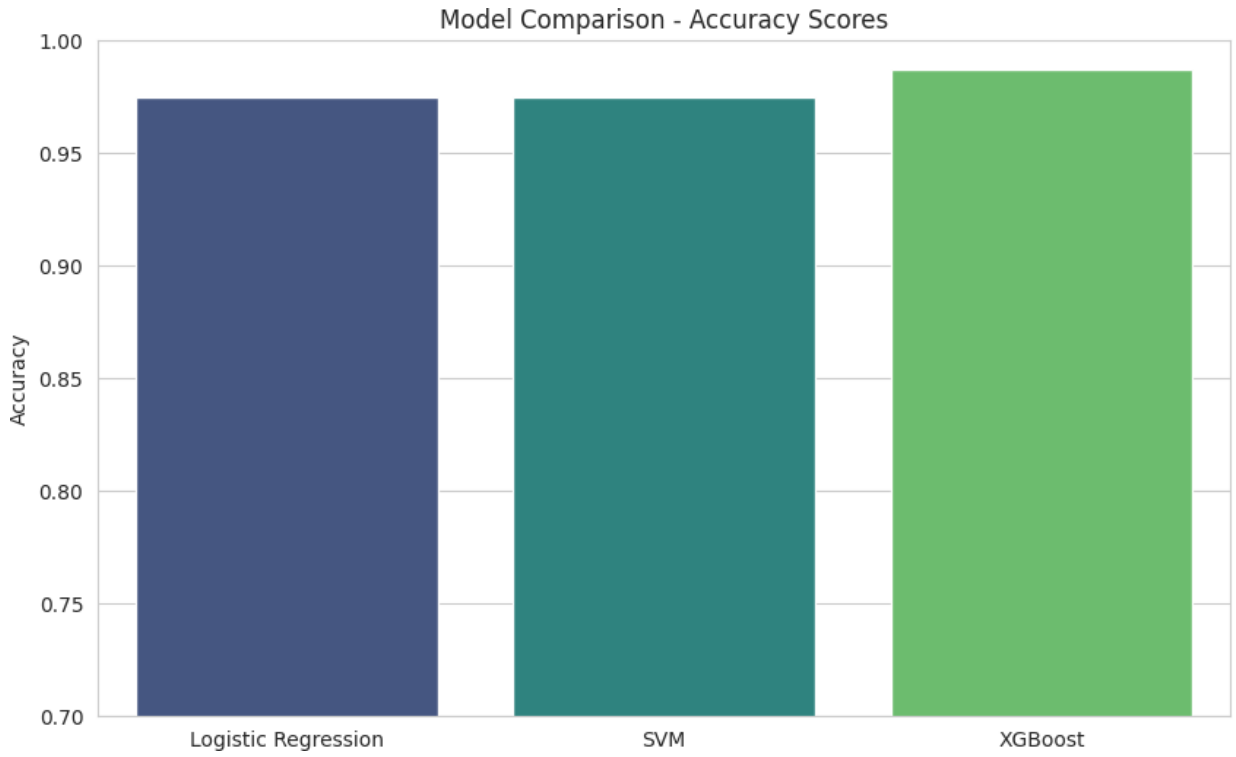
```
XGBoost Accuracy: 0.9875
```

```
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158:
UserWarning: [21:47:49] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
```

```
models = ['Logistic Regression', 'SVM', 'XGBoost']
accuracies = [accuracy_score(y_test, y_pred_lr),
              accuracy_score(y_test, y_pred_svm),
              accuracy_score(y_test, y_pred_xgb)]

plt.figure(figsize=(10, 6))
sns.barplot(x=models, y=accuracies, palette='viridis')
plt.title('Model Comparison - Accuracy Scores')
plt.ylabel('Accuracy')
plt.ylim(0.7, 1.0)
plt.show()
```

```
<ipython-input-96-7dcec0b39e18>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.
```

```
  sns.barplot(x=models, y=accuracies, palette='viridis')
```




```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay

# Re-train with reduced number of estimators for faster execution
model = XGBClassifier(n_estimators=100, use_label_encoder=False,
eval_metric='logloss')
model.fit(X_train_scaled, y_train)

# Predictions
y_pred = model.predict(X_test_scaled)

# Accuracy
accuracy = accuracy_score(y_test, y_pred)

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap=plt.cm.Blues)
plt.title(f'XGBoost Confusion Matrix (Accuracy: {accuracy*100:.2f}%)')
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158:
UserWarning: [21:49:56] WARNING: /workspace/src/learner.cc:740:
```

```
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
```