Healthcare Data analysis

Code:

```
# ==================== Import Libraries ====================

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import shap

import plotly.express as px

import joblib

import warnings


from sklearn.model_selection import train_test_split, cross_val_score

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.cluster import KMeans

from sklearn.decomposition import PCA

from yellowbrick.cluster import KElbowVisualizer


warnings.filterwarnings('ignore')


# ==================== Load & Clean Data ====================

df = pd.read_csv("/content/healthcare_dataset.csv.zip")

df.dropna(inplace=True)

df = df[df['Age'] < 100]
```

```
print("Columns:", df.columns)

# ========== Encode Categorical Columns ==========
categoricals = ['Gender', 'Blood Type', 'Medication']
le_dict = {}
for col in categoricals:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    le_dict[col] = le

# ========== Scale Numerical Features ==========
scaler = StandardScaler()
df[['Age', 'Billing Amount']] = scaler.fit_transform(df[['Age', 'Billing Amount']])

# ========== Correlation Heatmap ==========
df_numeric = df.drop(columns=['Name', 'Medical Condition', 'Discharge Date'], errors='ignore')
numeric_cols = df_numeric.select_dtypes(include=np.number).columns
plt.figure(figsize=(10,6))
sns.heatmap(df_numeric[numeric_cols].corr(), annot=True, cmap="coolwarm")
plt.title("Feature Correlation")
plt.show()

# ========== Boxplots ==========
plt.figure(figsize=(10,6))
sns.boxplot(x="Blood Type", y="Billing Amount", data=df)
plt.title("Billing Amount by Blood Type")
plt.show()

plt.figure(figsize=(10,6))
sns.boxplot(x="Medication", y="Billing Amount", data=df)
```

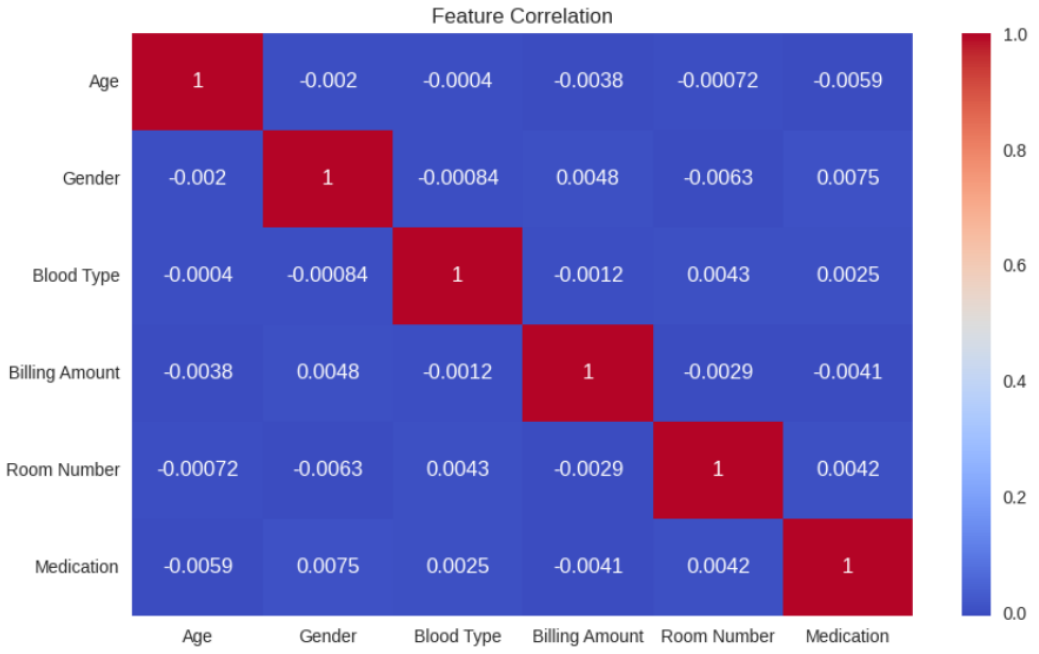
```
plt.title("Billing Amount by Medication")
plt.xticks(rotation=45)
plt.show()

# ========== Date Columns ==========
df['Date of Admission'] = pd.to_datetime(df['Date of Admission'])
df['Discharge Date'] = pd.to_datetime(df['Discharge Date'])
df['stay_duration'] = (df['Discharge Date'] - df['Date of Admission']).dt.days

# ========== Clustering: KMeans ==========
features = df[['Age', 'Billing Amount', 'stay_duration']]
model = KMeans()
elbow = KElbowVisualizer(model, k=(2, 10))
elbow.fit(features)
elbow.show()

# Apply Best K
k_best = elbow.elbow_value_
kmeans = KMeans(n_clusters=k_best)
df['cluster'] = kmeans.fit_predict(features)

# PCA Cluster Visualization
pca = PCA(n_components=2)
df_pca = pca.fit_transform(features)
plt.scatter(df_pca[:, 0], df_pca[:, 1], c=df['cluster'], cmap='tab10')
plt.title("Patient Clusters")
plt.xlabel("PCA1")
plt.ylabel("PCA2")
plt.show()

# ========== Train-Test Split ==========
```

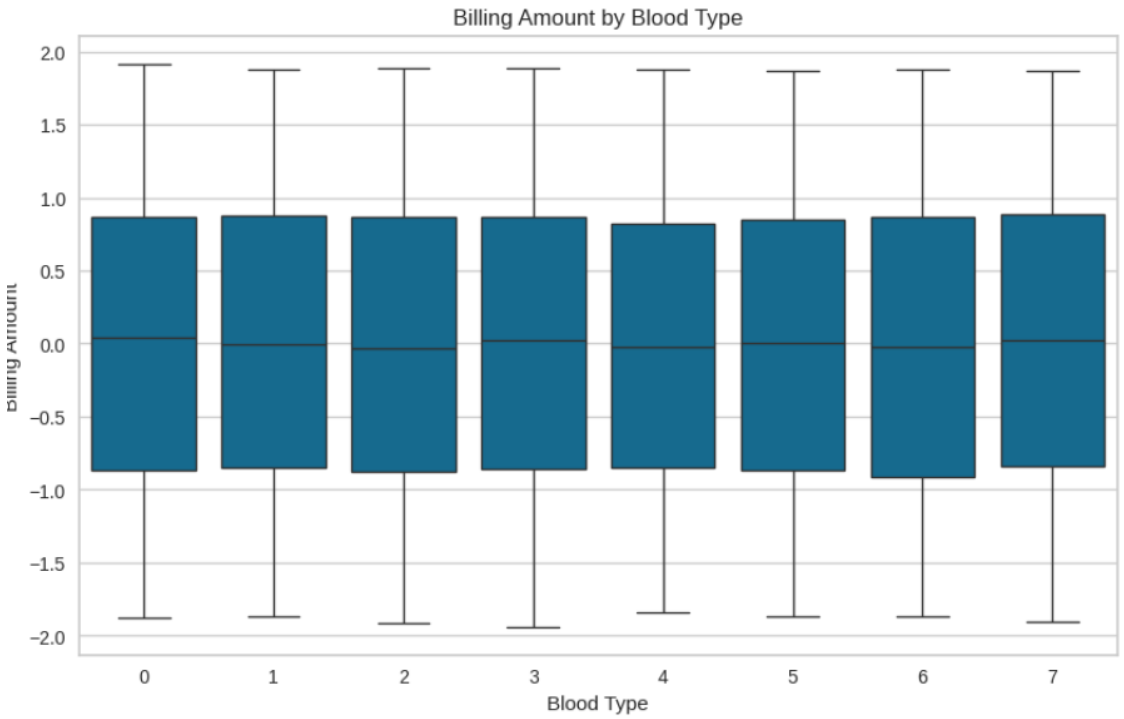
```
X = df[['Age', 'Gender', 'Blood Type', 'Medication', 'stay_duration', 'Billing Amount']]
y = df['Insurance Provider']

# Encode Target Variable
le_y = LabelEncoder()
y_encoded = le_y.fit_transform(y)

X_train, X_test, y_train_encoded, y_test_encoded = train_test_split(X, y_encoded, test_size=0.2, random_state=42)

# ========== Model Training ==========
model = GradientBoostingClassifier()
model.fit(X_train, y_train_encoded)

# ========== Evaluation ==========
y_pred_encoded = model.predict(X_test)
print(classification_report(y_test_encoded, y_pred_encoded))
y_pred_proba = model.predict_proba(X_test)
print("ROC AUC Score:", roc_auc_score(y_test_encoded, y_pred_proba, multi_class='ovr'))

sns.heatmap(confusion_matrix(y_test_encoded, y_pred_encoded), annot=True, fmt='d', cmap="YlGnBu")
plt.title("Confusion Matrix")
plt.show()

# ========== SHAP Explainability ==========
explainer = shap.Explainer(model.predict_proba, X_train)
shap_values = explainer(X_test)
shap.summary_plot(shap_values, X_test)

# ========== Cross Validation ==========
```

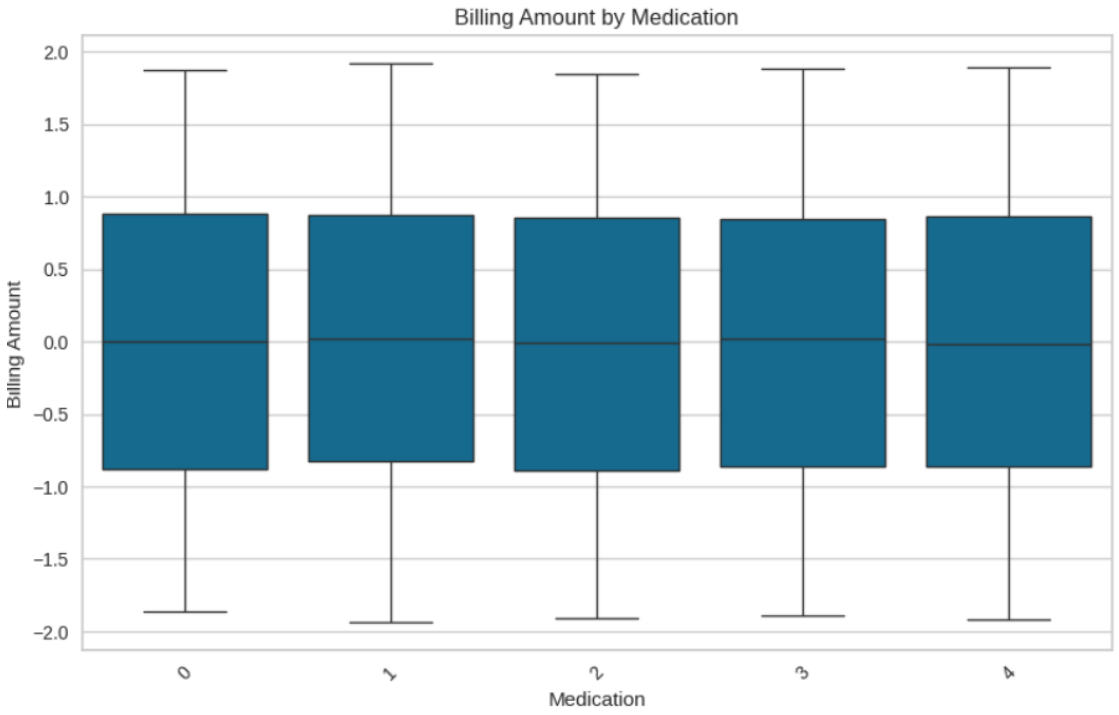
```
scores = cross_val_score(model, X, y_encoded, cv=5, scoring='accuracy')
print("Cross-Validation Accuracy:", scores.mean())

# ========== Feature Importance ==========
importances = pd.Series(model.feature_importances_, index=X.columns)
importances.sort_values().plot(kind='barh')
plt.title("Feature Importance")
plt.show()

# ========== Save Model ==========
joblib.dump(model, "readmission_model.pkl")
joblib.dump(le_dict, "label_encoders.pkl")

# ========== Plotly Visualization ==========
fig = px.scatter(df, x="Age", y="Billing Amount", color="cluster", title="Patient Clusters (Age vs Billing Amount)")
fig.show()
```

Output:

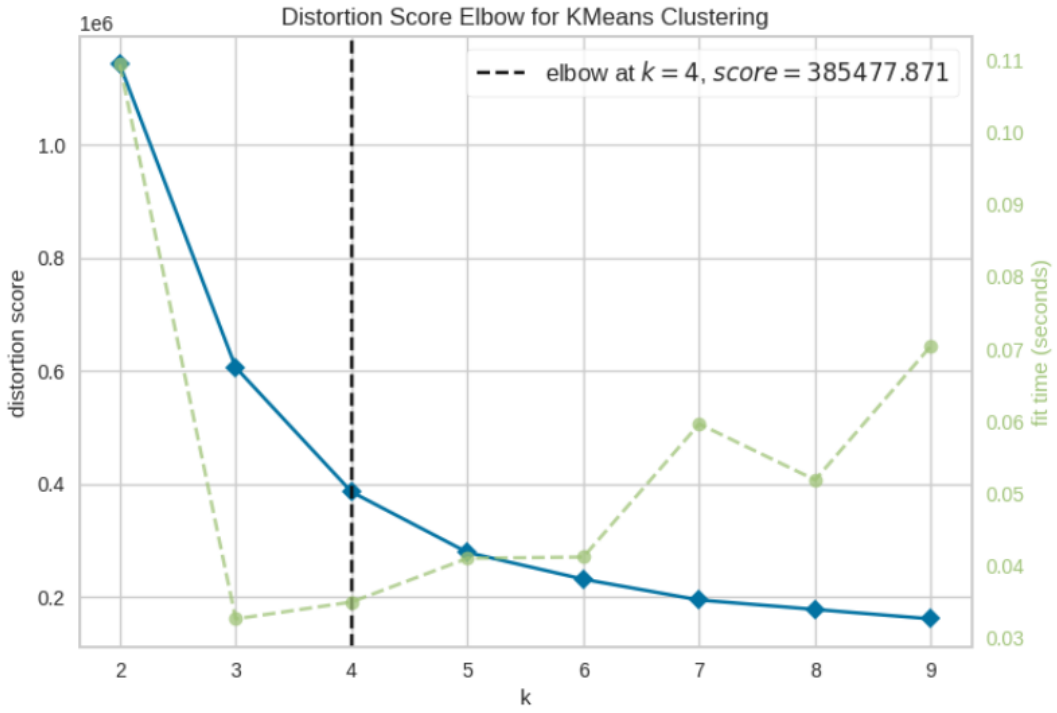
```
Columns in df_numeric before correlation calculation:
Index(['Age', 'Gender', 'Blood Type', 'Date of Admission', 'Doctor',
       'Hospital', 'Insurance Provider', 'Billing Amount', 'Room Number',
       'Admission Type', 'Medication', 'Test Results'],
      dtype='object')
```

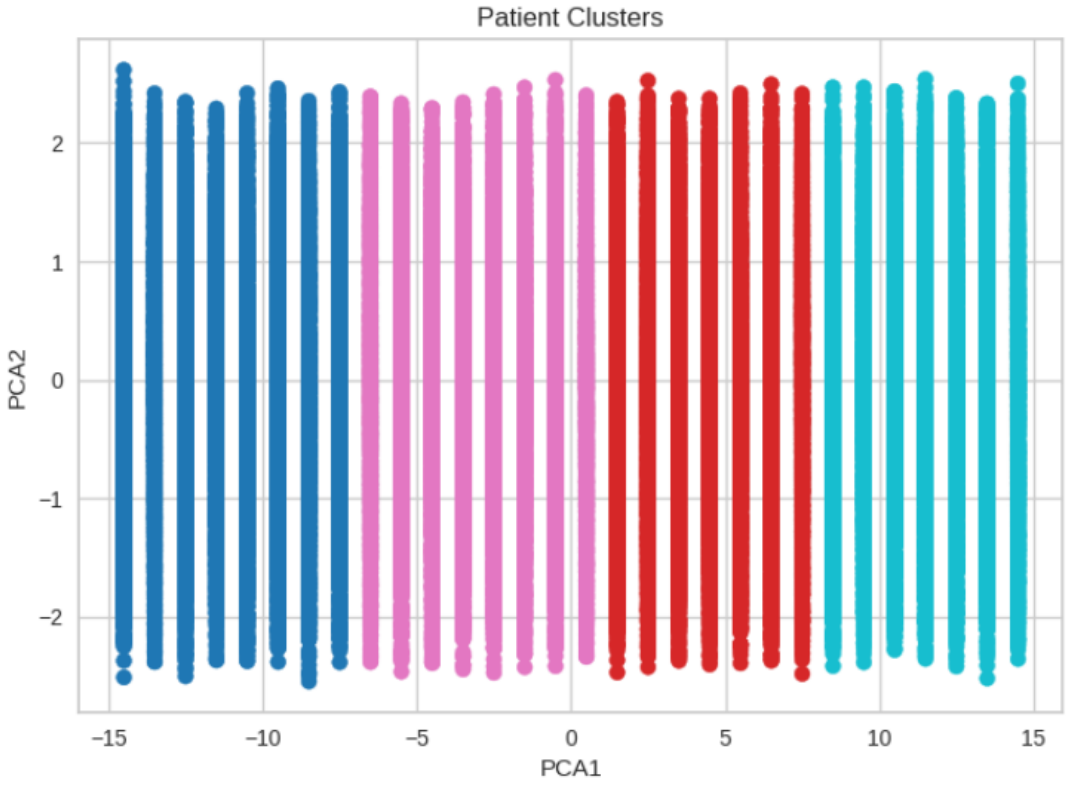






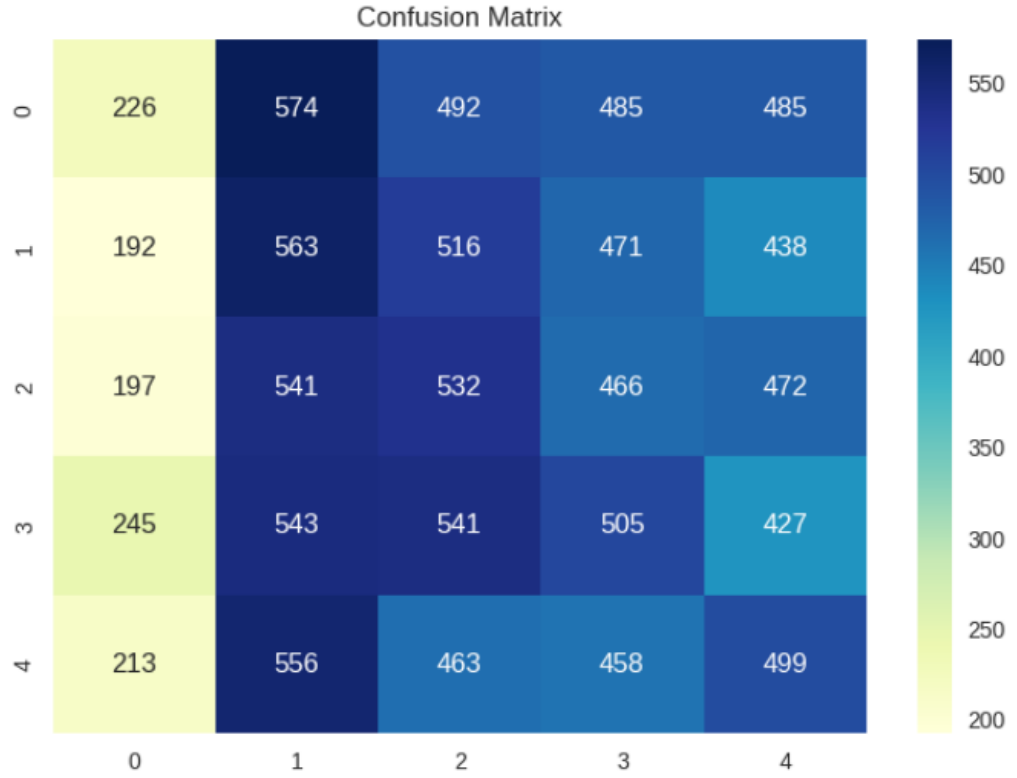

```
Columns in df before converting dates:
Index(['Name', 'Age', 'Gender', 'Blood Type', 'Medical Condition',
       'Date of Admission', 'Doctor', 'Hospital', 'Insurance Provider',
       'Billing Amount', 'Room Number', 'Admission Type', 'Discharge Date',
       'Medication', 'Test Results'],
      dtype='object')
```



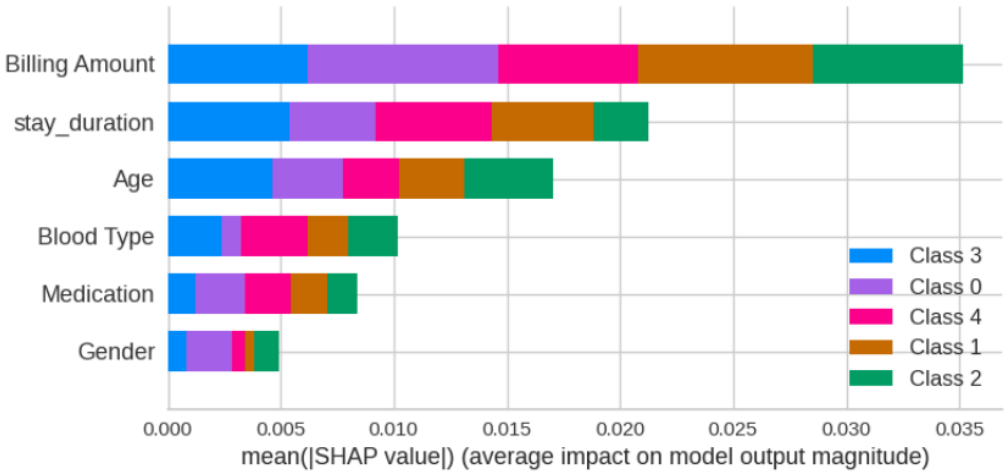

Patient Clusters
PCA2
PCA1
2
1
0
−1
−2
−15
−10
−5
0
5
10
15

	precision	recall	f1-score	support
0	0.21	0.10	0.14	2262
1	0.20	0.26	0.23	2180
2	0.21	0.24	0.22	2208
3	0.21	0.22	0.22	2261
4	0.21	0.23	0.22	2189
accuracy			0.21	11100
macro avg	0.21	0.21	0.21	11100
weighted avg	0.21	0.21	0.20	11100

ROC AUC: 0.5160712119723072



ExactExplainer explainer: 11101it [07:48, 23.48it/s]



```
Cross-Validation Accuracy:  0.2101801801801802
```