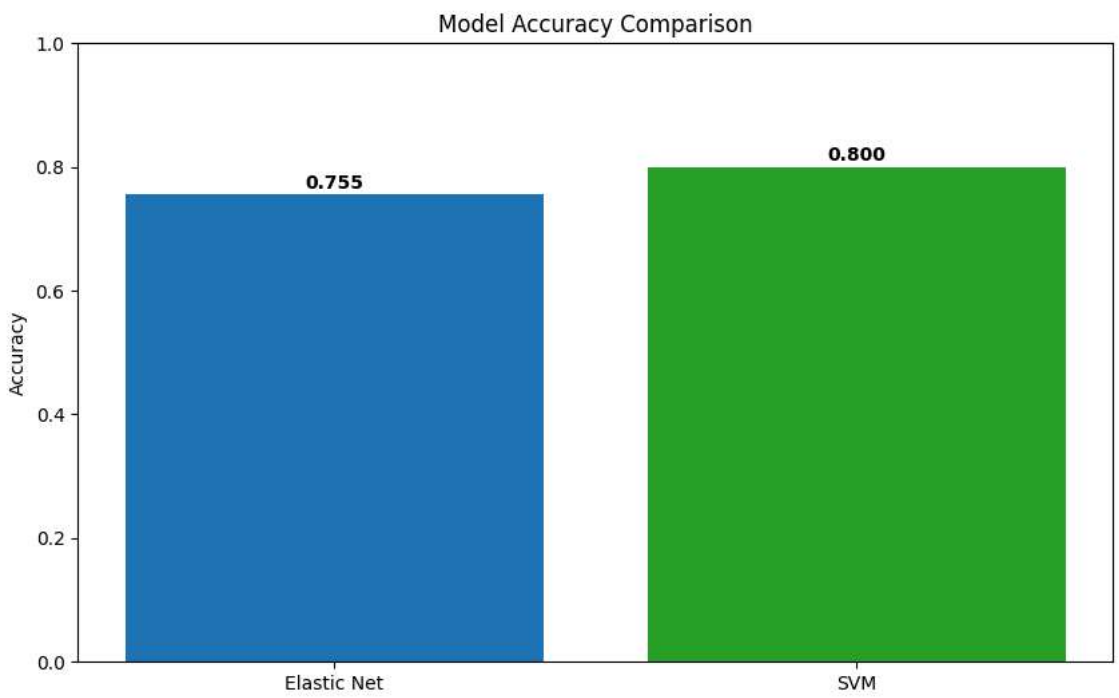
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, roc_auc_score

# Load dataset
df = pd.read_csv('finalDementia.csv')

# Preprocessing
# Encode categorical columns into numeric
df['Gender'] = LabelEncoder().fit_transform(df['Gender'])
df['Depression_Status'] = LabelEncoder().fit_transform(df['Depression_Status'])
df['Medication_History'] = LabelEncoder().fit_transform(df['Medication_History'])
df['Chronic_Health_Conditions'] = LabelEncoder().fit_transform(df['Chronic_Health_Conditions'])
df['CCI'] = pd.to_numeric(df['CCI'], errors='coerce')

# Create a 'Mortality' column based on assumptions
df['Mortality'] = (
    ((df['Dementia'] == 1) & (df['Heart Failure'] == 1)) | (df['Age'] > 80) &
    (df['MMSE'] < 12) & (df['CCI'] >= 3) & (df['Gender'] == 1)
).astype(int)

# Convert all columns to numeric, forcing errors into NaN
X = df.drop('Mortality', axis=1).apply(pd.to_numeric, errors='coerce')

# Handle missing values if necessary (e.g., replace NaN with the column mean)
X.fillna(X.mean(), inplace=True)

y = df['Mortality']

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scale features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Apply PCA for feature extraction
pca = PCA(n_components=10)  # Reduce to 10 principal components
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)

print("Explained Variance Ratio by PCA components:", pca.explained_variance_ratio_)

# Elastic Net-regularized Logistic Regression
elastic_net = LogisticRegression(penalty='elasticnet', solver='saga', l1_ratio=0.5, max_iter=10000)
elastic_net.fit(X_train_pca, y_train)

# Support Vector Machine (SVM)
svm = SVC(probability=True)  # Set probability=True for ROC-AUC score calculation
svm.fit(X_train_pca, y_train)

# Predictions
y_pred_en = elastic_net.predict(X_test_pca)
y_pred_svm = svm.predict(X_test_pca)

# Evaluation for Elastic Net
print("Elastic Net Regularized Logistic Regression")
print(f"Accuracy: {accuracy_score(y_test, y_pred_en)}")
print(f"Classification Report: \n{classification_report(y_test, y_pred_en, zero_division=0)}")
print(f"AUC-ROC: {roc_auc_score(y_test, elastic_net.predict_proba(X_test_pca)[:, 1])}")

# Evaluation for SVM
print("Support Vector Machine (SVM)")
print(f"Accuracy: {accuracy_score(y_test, y_pred_svm)}")
print(f"Classification Report: \n{classification_report(y_test, y_pred_svm, zero_division=0)}")
print(f"AUC-ROC: {roc_auc_score(y_test, svm.predict_proba(X_test_pca)[:, 1])}")
```

```
Explained Variance Ratio by PCA components: [0.07355017 0.06615277 0.05851869 0.04925935 0.04547888 0.04366749
 0.04291123 0.04195098 0.04083397 0.03909465]
Elastic Net Regularized Logistic Regression
```

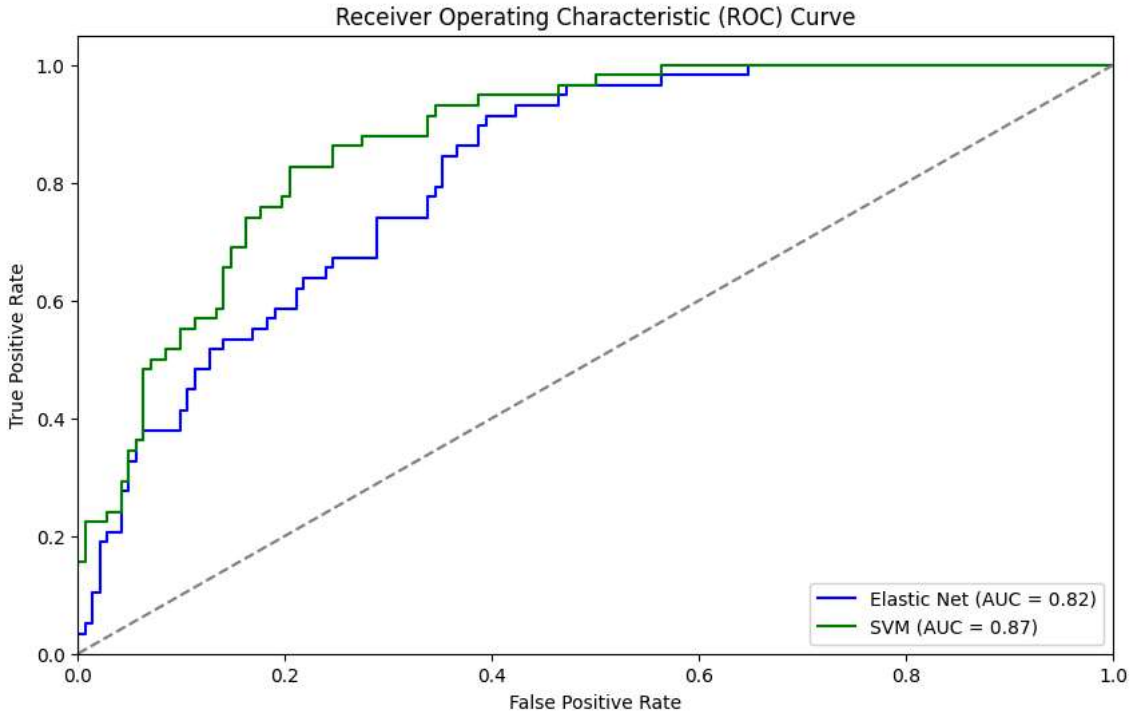
```
Accuracy: 0.755
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.90      0.84       142
           1       0.62      0.40      0.48        58

    accuracy                           0.76       200
   macro avg       0.70      0.65      0.66       200
weighted avg       0.74      0.76      0.74       200

AUC-ROC: 0.8153229723166586
Support Vector Machine (SVM)
Accuracy: 0.8
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.92      0.87       142
           1       0.72      0.50      0.59        58

    accuracy                           0.80       200
   macro avg       0.77      0.71      0.73       200
weighted avg       0.79      0.80      0.79       200

AUC-ROC: 0.8712967459932006
```

```
import matplotlib.pyplot as plt
import numpy as np

# Calculate accuracies
accuracies = [
    accuracy_score(y_test, y_pred_en),
    accuracy_score(y_test, y_pred_svm)
]

# Bar plot for accuracy
plt.figure(figsize=(10, 6))
plt.bar(['Elastic Net','SVM'], accuracies, color=['#1f77b4', '#2ca02c'])
plt.ylabel('Accuracy')
plt.title('Model Accuracy Comparison')
plt.ylim([0, 1])
for i, v in enumerate(accuracies):
    plt.text(i, v + 0.01, f"{v:.3f}", ha='center', fontweight='bold')
plt.show()
```

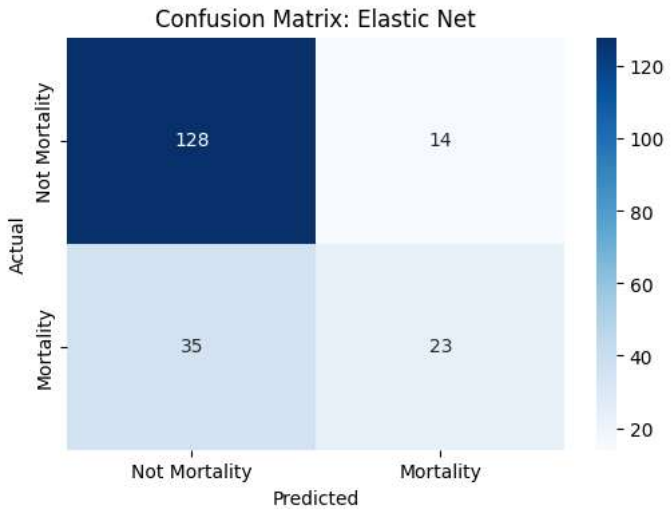


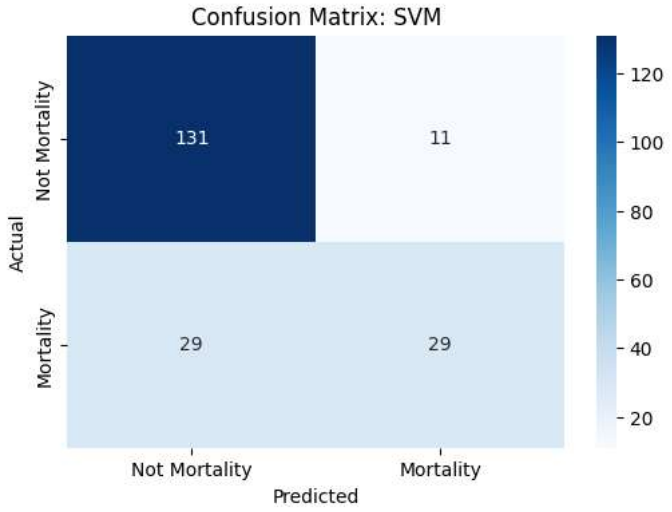

```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
```

```
# Calculate ROC curve for each model
# Use X_test_pca (transformed data) instead of X_test_scaled
fpr_en, tpr_en, _ = roc_curve(y_test, elastic_net.predict_proba(X_test_pca)[:, 1])
roc_auc_en = auc(fpr_en, tpr_en)

# Use X_test_pca (transformed data) instead of X_test_scaled
fpr_svm, tpr_svm, _ = roc_curve(y_test, svm.predict_proba(X_test_pca)[:, 1])
roc_auc_svm = auc(fpr_svm, tpr_svm)

# ROC curve plot
plt.figure(figsize=(10, 6))
plt.plot(fpr_en, tpr_en, color='blue', label=f'Elastic Net (AUC = {roc_auc_en:.2f})')
plt.plot(fpr_svm, tpr_svm, color='green', label=f'SVM (AUC = {roc_auc_svm:.2f})')
plt.plot([0, 1], [0, 1], color='grey', 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()
```

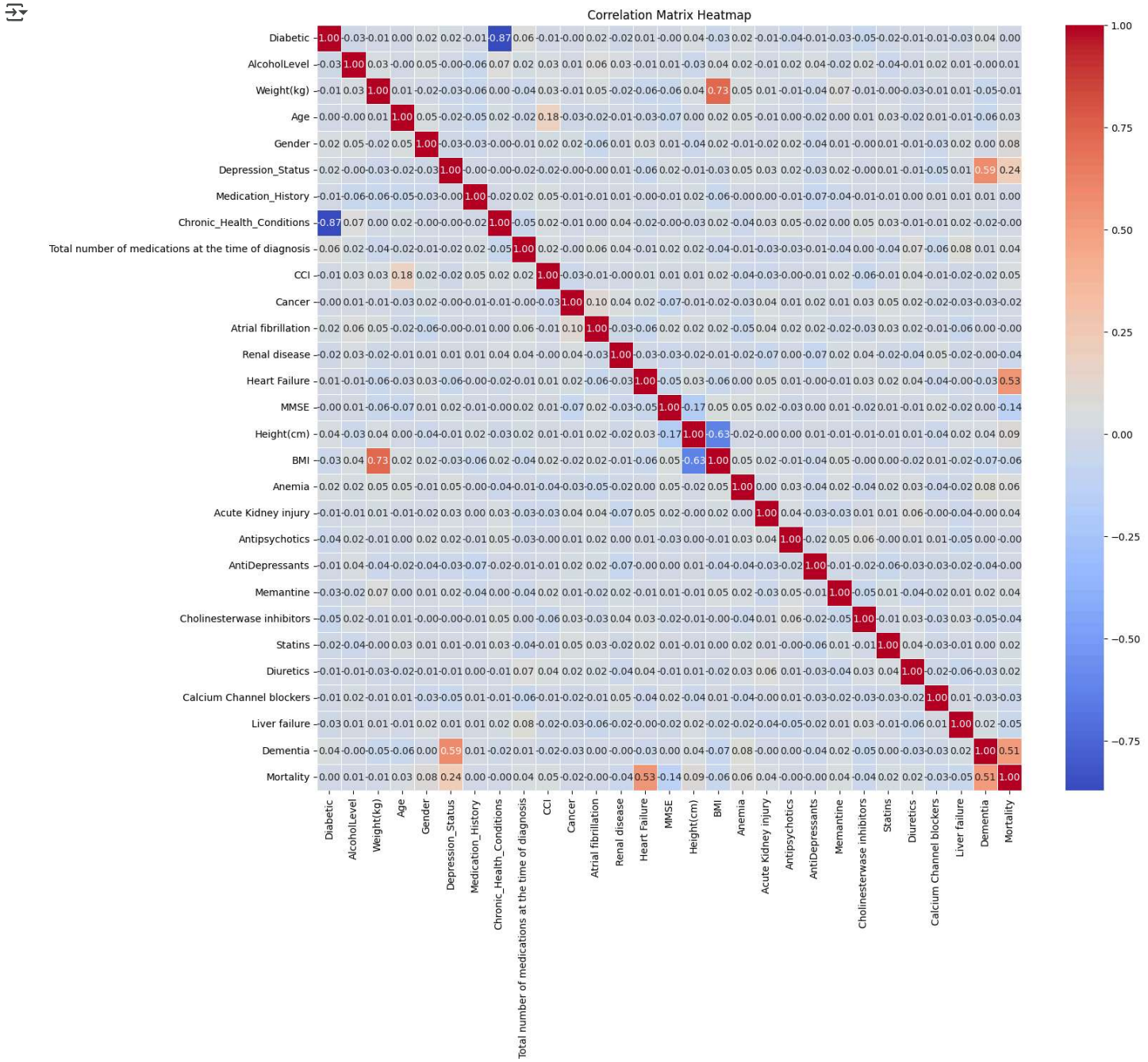



```
from sklearn.metrics import confusion_matrix
import seaborn as sns

def plot_confusion_matrix(y_true, y_pred, model_name):
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(6, 4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=['Not Mortality', 'Mortality'],
                yticklabels=['Not Mortality', 'Mortality'])
    plt.ylabel('Actual')
    plt.xlabel('Predicted')
    plt.title(f'Confusion Matrix: {model_name}')
    plt.show()

# Plot confusion matrix for each model
plot_confusion_matrix(y_test, y_pred_en, 'Elastic Net')
plot_confusion_matrix(y_test, y_pred_svm, 'SVM')
```






```
# Select only numeric columns from the DataFrame
numeric_df = df.select_dtypes(include='number')

# Calculate the correlation matrix for numeric columns
corr_matrix = numeric_df.corr()

# Plot the heatmap
plt.figure(figsize=(16, 14))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title("Correlation Matrix Heatmap")
plt.show()
```




```
from sklearn.ensemble import StackingClassifier
from sklearn.model_selection import cross_val_score

# Define base learners
base_learners = [
    ('elastic_net', LogisticRegression(penalty='elasticnet', solver='saga', l1_ratio=0.5, max_iter=10000)),
    ('svm', SVC(probability=True))
]

# Define stacking ensemble (Logistic Regression as meta-learner)
stacked_model = StackingClassifier(estimators=base_learners, final_estimator=LogisticRegression())

# Fit the stacked model
stacked_model.fit(X_train_scaled, y_train)

# Predictions
y_pred_stack = stacked_model.predict(X_test_scaled)

# Evaluation
```

```
print("Stacking Classifier")
print(f"Accuracy: {accuracy_score(y_test, y_pred_stack)}")
print(f"Classification Report: \n{classification_report(y_test, y_pred_stack,zero_division=0)}")
print(f"AUC-ROC: {roc_auc_score(y_test, stacked_model.predict_proba(X_test_scaled)[:, 1])}")
```

```
Stacking Classifier
Accuracy: 0.97
Classification Report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       142
           1       1.00      0.90      0.95        58

    accuracy                           0.97       200
   macro avg       0.98      0.95      0.96       200
weighted avg       0.97      0.97      0.97       200

AUC-ROC: 0.9865225837785333
```

```
import pickle

# Assuming 'model' is your trained ML model
with open("model.pkl", "wb") as file:
    pickle.dump(stacked_model, file)
```