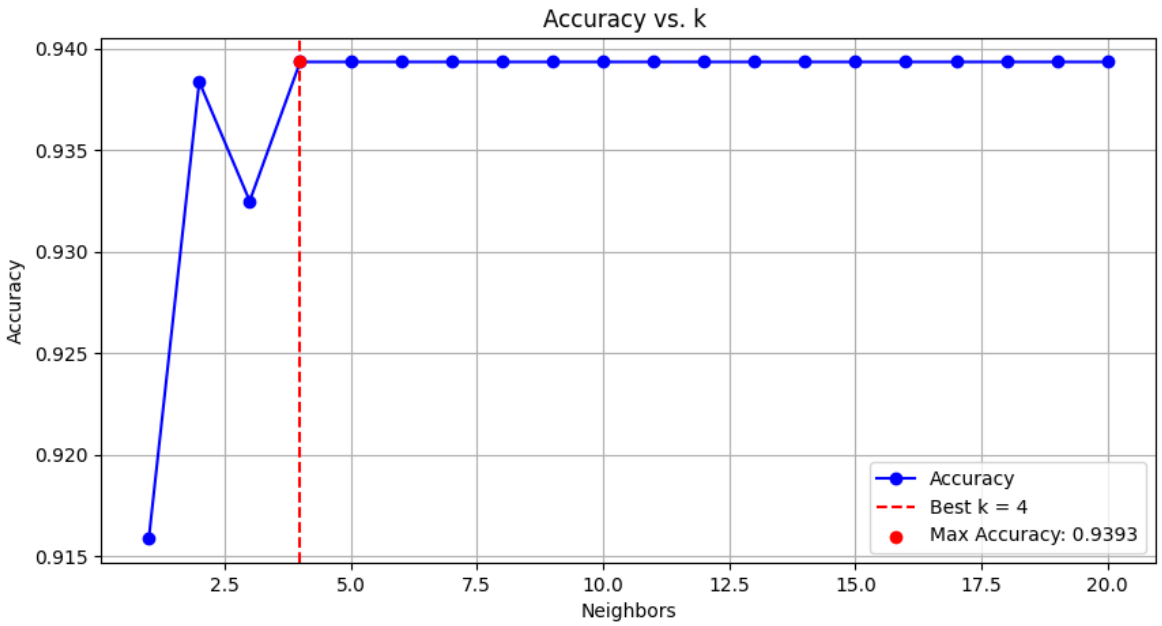
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

df = pd.read_csv(r"C:\Users\immad\Downloads\healthcare-dataset-stroke-
data.csv")

df['bmi'] = pd.to_numeric(df['bmi'], errors='coerce')  # Convert to numeric
df['bmi'].fillna(df['bmi'].mean(), inplace=True)  # Fill missing values with
column mean

categorical_cols = ['gender', 'ever_married', 'work_type', 'Residence_type',
'smoking_status']
label_encoders = {col: LabelEncoder().fit(df[col]) for col in
categorical_cols}
df[categorical_cols] = df[categorical_cols].apply(lambda col:
label_encoders[col.name].transform(col))

X = df.drop(columns=['id', 'stroke'])
y = df['stroke']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.2, random_state=42)

def evaluate_knn(k_values, X_train, X_test, y_train, y_test):
    accuracy_scores = []
    for k in k_values:
        model = KNeighborsClassifier(n_neighbors=k)
        model.fit(X_train, y_train)
        predictions = model.predict(X_test)
        accuracy_scores.append(accuracy_score(y_test, predictions))
    return accuracy_scores


k_values = range(1, 21)
accuracy_scores = evaluate_knn(k_values, X_train, X_test, y_train, y_test)


best_k = k_values[np.argmax(accuracy_scores)]
best_acc = max(accuracy_scores)


plt.figure(figsize=(10, 5))
plt.plot(k_values, accuracy_scores, marker='o', linestyle='-', color='b',
label="Accuracy")
plt.axvline(best_k, linestyle='--', color='r', label=f'Best k = {best_k}')
plt.scatter(best_k, best_acc, color='red', zorder=3, label=f'Max Accuracy:
{best_acc:.4f}')
plt.xlabel("Neighbors")
plt.ylabel("Accuracy")
```

```
plt.title("Accuracy vs. k ")
plt.legend()
plt.grid()
plt.show()

print(f"Optimal k: {best_k}, Achieved Accuracy: {best_acc * 100:.2f}%")
```

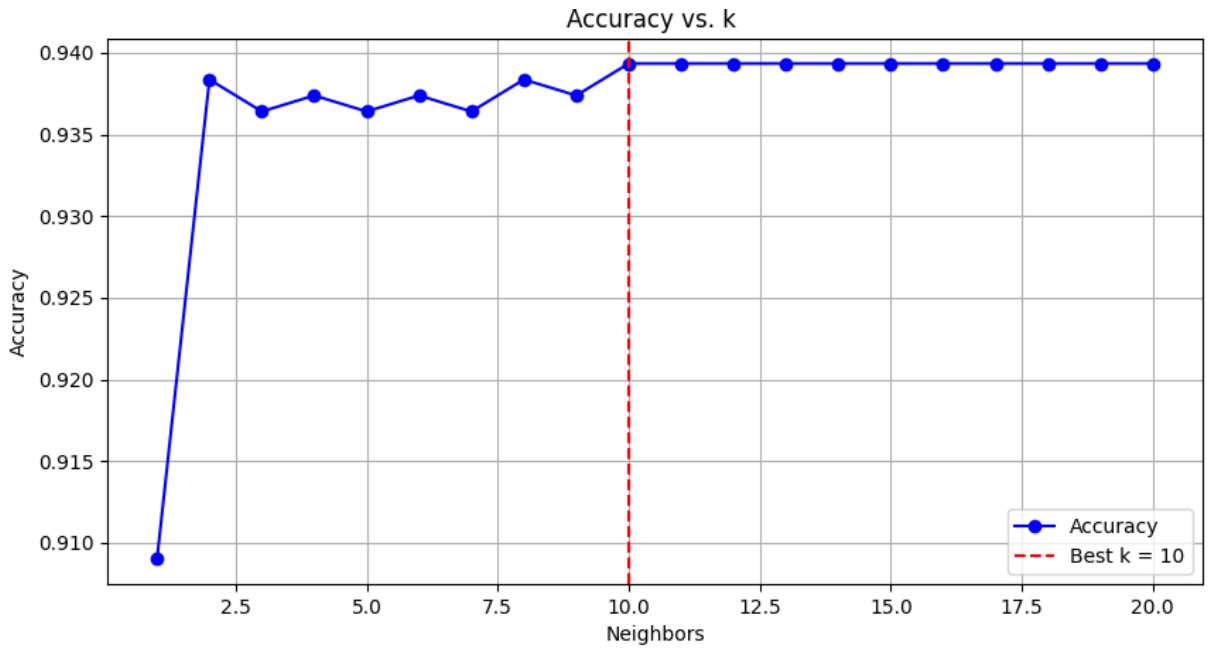



```
Optimal k: 4, Achieved Accuracy: 93.93%

Process finished with exit code 0
```

TASK 2:
The following features have been dropped;
ID *Unique identifier, irrelevant for prediction*
Gender *Weak correlation with stroke, minor effect compared to age and lifestyle factors)*
Ever Married *Marriage itself doesn't biologically contribute to stroke risk)*
Work Type *Does not directly indicate health risks; also encoded as categorical, which might not be effective for kNN*
Residence Type *Urban/Rural classification is too general to indicate meaningful health differences in this case*

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Load dataset
df = pd.read_csv(r"C:\Users\immad\Downloads\healthcare-dataset-stroke-
data.csv")  # Update with correct filename

df['bmi'] = pd.to_numeric(df['bmi'], errors='coerce')
df['bmi'].fillna(df.groupby('stroke')['bmi'].transform('mean'), inplace=True)

categorical_cols = ['smoking_status']  # Only keeping smoking_status as it's
relevant
label_encoders = {}
for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

selected_features = ['age', 'hypertension', 'heart_disease',
'avg_glucose_level', 'bmi', 'smoking_status']
X = df[selected_features]
y = df['stroke']

scaler = StandardScaler()
X = scaler.fit_transform(X)

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

k_values = range(1, 21)
accuracy_scores = []

for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    accuracy_scores.append(accuracy)

best_k = k_values[np.argmax(accuracy_scores)]
```

```
best_accuracy = max(accuracy_scores)

plt.figure(figsize=(10, 5))
plt.plot(k_values, accuracy_scores, marker='o', linestyle='-', color='b',
label="Accuracy")
plt.axvline(best_k, linestyle='--', color='r', label=f'Best k = {best_k}')
plt.xlabel("Neighbors")
plt.ylabel("Accuracy")
plt.title("Accuracy vs. k ")
plt.legend()
plt.grid()
plt.show()

print(f"Best k: {best_k} with Accuracy: {best_accuracy*100:.4f}%")
```

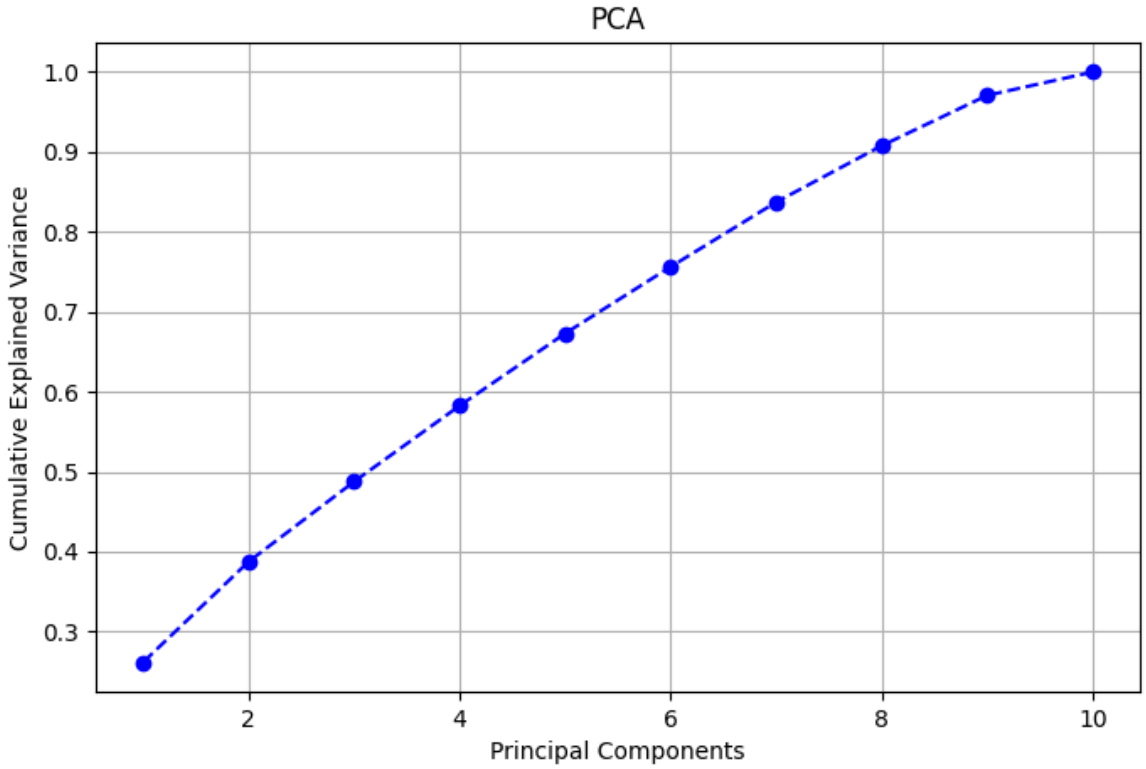



```
Best k: 10 with Accuracy: 93.9335%

Process finished with exit code 0
```

TASK 3

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.decomposition import PCA
from sklearn.metrics import accuracy_score

df = pd.read_csv(r"C:\Users\immad\Downloads\healthcare-dataset-stroke-
data.csv")

df['bmi'] = pd.to_numeric(df['bmi'], errors='coerce')
df['bmi'].fillna(df.groupby('stroke')['bmi'].transform('mean'), inplace=True)

categorical_cols = ['gender', 'ever_married', 'work_type', 'Residence_type',
'smoking_status']
label_encoders = {}
for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

X = df.drop(columns=['id', 'stroke'])
y = df['stroke']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

pca = PCA(n_components=X_scaled.shape[1])
X_pca = pca.fit_transform(X_scaled)

explained_variance = np.cumsum(pca.explained_variance_ratio_)

plt.figure(figsize=(8, 5))
plt.plot(range(1, X_scaled.shape[1] + 1), explained_variance, marker='o',
linestyle='--', color='b')
plt.xlabel('Principal Components')
plt.ylabel('Cumulative Explained Variance')
plt.title('PCA')
plt.grid()
plt.show()

n_components_optimal = np.argmax(explained_variance >= 0.95) + 1
if n_components_optimal == 0:
    n_components_optimal = X_scaled.shape[1]

print(f'Optimal number of PCA components: {n_components_optimal}')

pca = PCA(n_components=n_components_optimal)
X_pca_reduced = pca.fit_transform(X_scaled)

X_train, X_test, y_train, y_test = train_test_split(X_pca_reduced, y,
test_size=0.2, random_state=42)

knn = KNeighborsClassifier(n_neighbors=8)
knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)
accuracy_pca = accuracy_score(y_test, y_pred)
print(f'Accuracy after PCA: {accuracy_pca:.4f}')
```

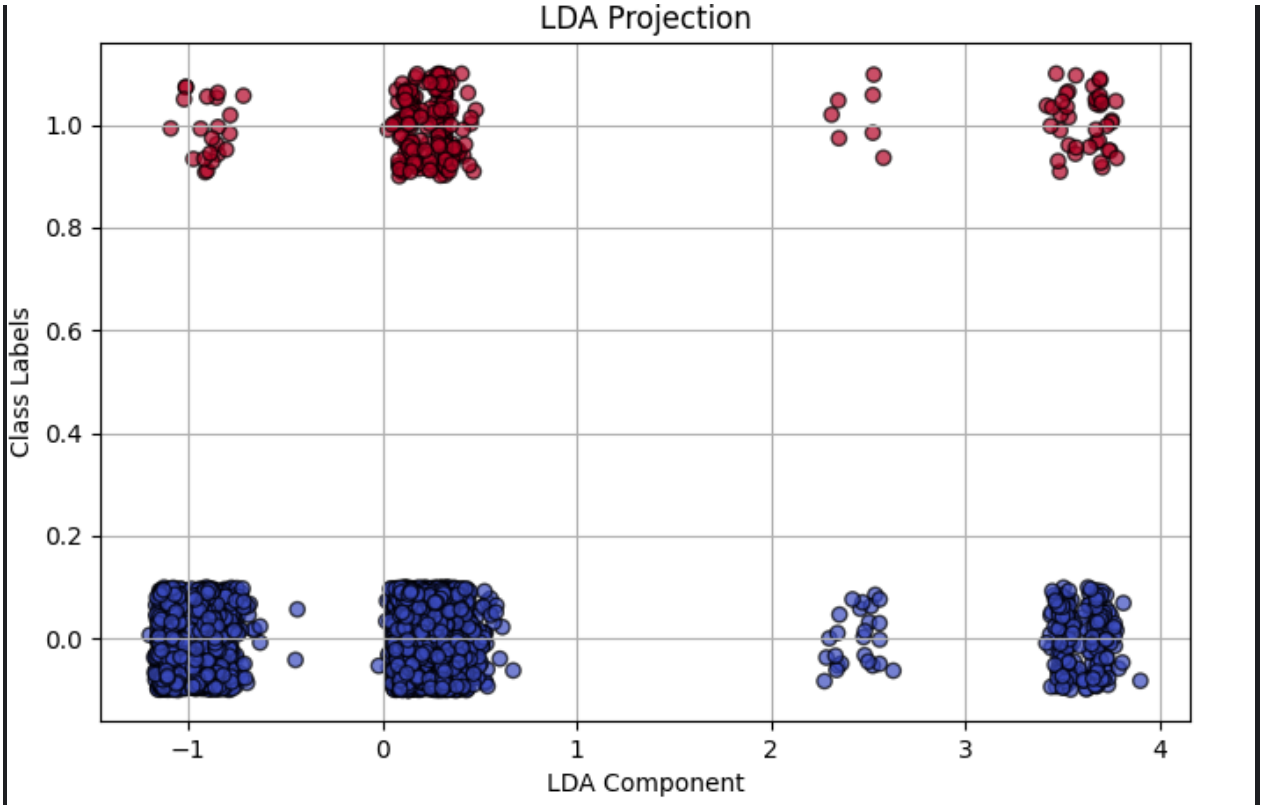



```
Optimal number of PCA components: 9
Accuracy after PCA: 0.9393

Process finished with exit code 0
```

```
TASK 4:

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA


df = pd.read_csv(r"C:\Users\immad\Downloads\healthcare-dataset-stroke-
data.csv")

df['bmi'] = pd.to_numeric(df['bmi'], errors='coerce')
df['bmi'].fillna(df.groupby('stroke')['bmi'].transform('mean'), inplace=True)

categorical_cols = ['ever_married', 'Residence_type']
label_encoders = {}
for col in categorical_cols:
    le = LabelEncoder()
```

```
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

selected_features = ['heart_disease', 'ever_married', 'Residence_type',
'bmi']
X_selected = df[selected_features]
y = df['stroke']  # Target variable

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_selected)

lda = LDA(n_components=1)
X_lda = lda.fit_transform(X_scaled, y)

X_train, X_test, y_train, y_test = train_test_split(X_lda, y, test_size=0.2,
random_state=42)

knn = KNeighborsClassifier(n_neighbors=8)
knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)
accuracy_lda = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy_lda:.4f}')

plt.figure(figsize=(8, 5))
plt.scatter(X_lda, y + np.random.uniform(-0.1, 0.1, size=len(y)), c=y,
cmap='coolwarm', edgecolors='k', alpha=0.7)
plt.xlabel('LDA Component')
plt.ylabel('Class Labels')
plt.title('LDA Projection')
plt.grid()
plt.show()
```

```
Accuracy: 0.9472
```



Dimensions reduced without loss of info.
using important features accuracy increased.
TASK 5:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA

df = pd.read_csv(r"C:\Users\immad\Downloads\healthcare-dataset-stroke-
data.csv")

df['bmi'] = pd.to_numeric(df['bmi'], errors='coerce')
df['bmi'].fillna(df.groupby('stroke')['bmi'].transform('mean'), inplace=True)

label_encoders = {}
for col in ['ever_married', 'Residence_type']:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

top_4_features = ['heart_disease', 'ever_married', 'Residence_type', 'bmi']
X_selected = df[top_4_features]
y = np.array(df['stroke'])
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_selected)

lda = LDA(n_components=1)
X_lda = lda.fit_transform(X_scaled, y)

accuracies = []
for i in range(5):
    X_train, X_test, y_train, y_test = train_test_split(X_lda, y,
test_size=0.3, stratify=y, random_state=i)

    knn = KNeighborsClassifier(n_neighbors=8)
    knn.fit(X_train, y_train)

    y_pred = knn.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    accuracies.append(accuracy)
    print(f'Run {i + 1}: Accuracy = {accuracy:.4f}')

mean_accuracy = np.mean(accuracies)
std_accuracy = np.std(accuracies)
print(f'\nAvg Accuracy: {mean_accuracy:.4f}')
print(f'Standard Deviation: {std_accuracy:.4f}')
```

```
Run 1: Accuracy = 0.9537
Run 2: Accuracy = 0.9537
Run 3: Accuracy = 0.9569
Run 4: Accuracy = 0.9504
Run 5: Accuracy = 0.9524

Avg Accuracy: 0.9534
Standard Deviation: 0.0021
```