1. Import libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.metrics import (confusion_matrix, classification_report, accuracy_score,
                             precision_score, recall_score, f1_score)
```

2. Read Dataset (simulate data for this example) np.random.seed(42)

```
# 3. Read Dataset (simulate data for this example)
np.random.seed(42)
df = pd.DataFrame({
    'Age': np.random.randint(20, 80, 1000),
    'BMI': np.random.uniform(18, 35, 1000),
    'BloodPressure': np.random.randint(90, 180, 1000),
    'Glucose': np.random.randint(70, 200, 1000),
    'Gender': np.random.choice(['Male', 'Female'], 1000),
    'Geography': np.random.choice(['Urban', 'Rural'], 1000),
    'Smoker': np.random.choice([0, 1], 1000),
    'FamilyHistory': np.random.choice([0, 1], 1000),
    'DiseasePresent': np.random.choice([0, 1], 1000, p=[0.7, 0.3])
})
```

3. exploratory data analysis

```
# 4.1 Shape
print("Dataset Shape:", df.shape)

# 4.2 Preview
print(df.head())

# 4.3 Summary
print(df.info())

# 4.4 Statistical Properties
print(df.describe())
```

```
Dataset Shape: (1000, 9)
   Age        BMI  BloodPressure  Glucose  Gender Geography  Smoker  \
0   58  18.797240            173      145  Female     Urban       1
1   71  22.567432            157      145    Male     Rural       1
2   48  18.377141             96      116  Female     Urban       1
3   34  26.468808            143       84  Female     Rural       1
4   62  26.095582             90      174  Female     Rural       1

   FamilyHistory  DiseasePresent
0              0               0
1              0               0
2              1               0
3              1               0
4              1               0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Age             1000 non-null   int64
 1   BMI             1000 non-null   float64
 2   BloodPressure   1000 non-null   int64
 3   Glucose         1000 non-null   int64
 4   Gender          1000 non-null   object
 5   Geography       1000 non-null   object
 6   Smoker          1000 non-null   int64
 7   FamilyHistory   1000 non-null   int64
```

```
 8   DiseasePresent  1000 non-null   int64  
dtypes: float64(1), int64(6), object(2)
memory usage: 70.4+ KB
None
               Age          BMI  BloodPressure      Glucose       Smoker  \
count  1000.000000  1000.000000    1000.000000  1000.000000  1000.000000   
mean     50.200000    26.709609     132.908000   135.280000     0.484000   
std      17.372905     4.827534      26.305542    36.699434     0.499994   
min      20.000000    18.004038      90.000000    70.000000     0.000000   
25%      36.000000    22.637515     110.000000   104.750000     0.000000   
50%      51.000000    27.015266     131.000000   135.000000     0.000000   
75%      66.000000    30.790958     156.000000   168.000000     1.000000   
max      79.000000    34.989009     179.000000   199.000000     1.000000   

       FamilyHistory  DiseasePresent  
count    1000.000000     1000.000000  
mean        0.528000        0.290000  
std         0.499465        0.453989  
min         0.000000        0.000000  
25%         0.000000        0.000000  
50%         1.000000        0.000000  
75%         1.000000        1.000000  
max         1.000000        1.000000  
```

4. Feature Selection

```
features = df.drop('DiseasePresent', axis=1)
target = df['DiseasePresent']
```

5. Convert Categorical Columns to Numeric

```
# 6.1 Gender
df['Gender'] = LabelEncoder().fit_transform(df['Gender'])

# 6.2 Geography
df['Geography'] = LabelEncoder().fit_transform(df['Geography'])
```

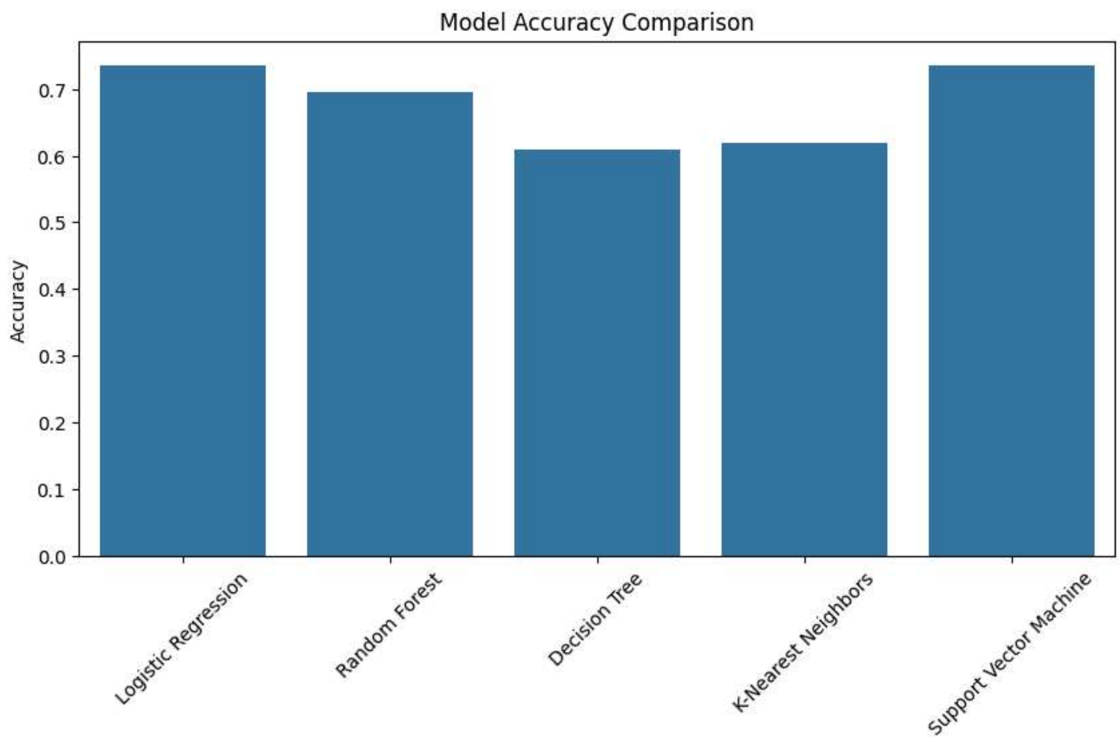
6. Feature Scaling

```
X = df.drop('DiseasePresent', axis=1)
y = df['DiseasePresent']

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

7. Model Training

```
# 8.1 Predict accuracy with different algorithms
models = {
    "Logistic Regression": LogisticRegression(),
    "Random Forest": RandomForestClassifier(),
    "Decision Tree": DecisionTreeClassifier(),
    "K-Nearest Neighbors": KNeighborsClassifier(),
    "Support Vector Machine": SVC()
}

accuracy_scores = {}
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    accuracy_scores[name] = acc
    print(f"{name} Accuracy: {acc:.2f}")

# 8.2 Plot classifier accuracy scores
plt.figure(figsize=(10, 5))
sns.barplot(x=list(accuracy_scores.keys()), y=list(accuracy_scores.values()))
plt.xticks(rotation=45)
plt.title("Model Accuracy Comparison")
plt.ylabel("Accuracy")
plt.show()
```

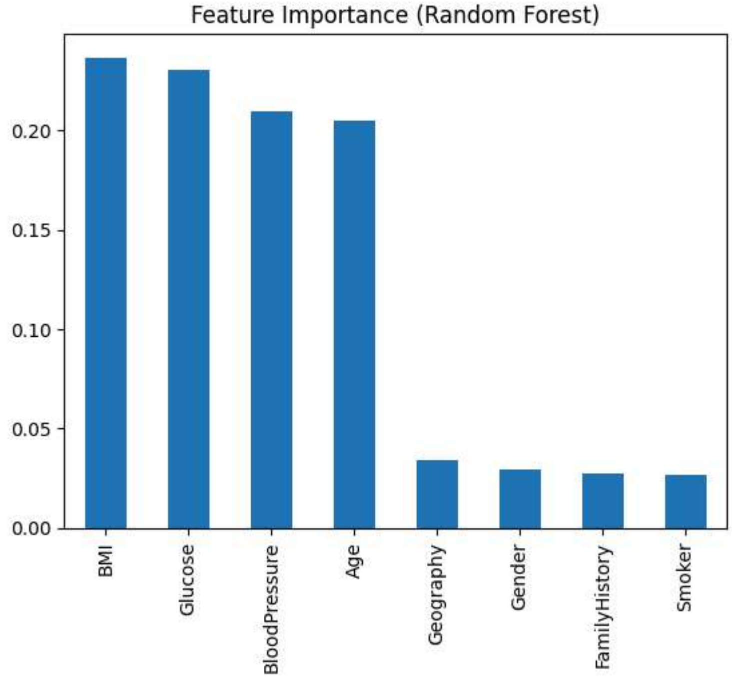
```
Logistic Regression Accuracy: 0.73
Random Forest Accuracy: 0.69
Decision Tree Accuracy: 0.61
K-Nearest Neighbors Accuracy: 0.62
Support Vector Machine Accuracy: 0.73
```



8. Feature Importance

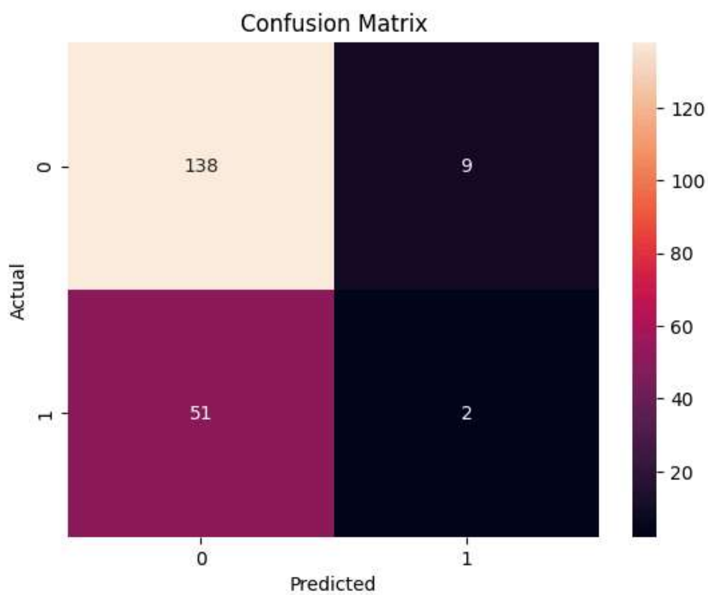
```
# 9.1 Using Random Forest
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
importances = rf.feature_importances_

# Plot
feature_names = df.drop('DiseasePresent', axis=1).columns
feat_df = pd.Series(importances, index=feature_names).sort_values(ascending=False)
feat_df.plot(kind='bar', title="Feature Importance (Random Forest)")
plt.show()
```

```
# 9.2 Drop least important feature (if needed)
# For demo, dropping the least important one
X_reduced = df.drop(columns=['DiseasePresent', feat_df.idxmin()])
X_reduced_scaled = scaler.fit_transform(X_reduced)
```



9. Confusion Matrix

```
y_pred_rf = rf.predict(X_test)
cm = confusion_matrix(y_test, y_pred_rf)
sns.heatmap(cm, annot=True, fmt='d')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



10. Classification Metrics

```
print("11.1 Classification Report:")
print(classification_report(y_test, y_pred_rf))

print("11.2 Accuracy:", accuracy_score(y_test, y_pred_rf))
```

```
print("11.3 Error:", 1 - accuracy_score(y_test, y_pred_rf))
print("11.4 Precision:", precision_score(y_test, y_pred_rf))
print("11.5 Recall:", recall_score(y_test, y_pred_rf))

# 11.6 TPR (Same as Recall)
tpr = recall_score(y_test, y_pred_rf)
print("11.6 True Positive Rate:", tpr)

# 11.7 FPR
fp = cm[0][1]
tn = cm[0][0]
fpr = fp / (fp + tn)
print("11.7 False Positive Rate:", fpr)

# 11.8 Specificity
specificity = tn / (tn + fp)
print("11.8 Specificity (TNR):", specificity)

# 11.9 F1 Score
print("11.9 F1 Score:", f1_score(y_test, y_pred_rf))

# 11.10 Support (from classification report)
```

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

           0       0.73      0.94      0.82       147
           1       0.18      0.04      0.06        53

    accuracy                           0.70       200
   macro avg       0.46      0.49      0.44       200
weighted avg       0.58      0.70      0.62       200

11.2 Accuracy: 0.7
11.3 Error: 0.30000000000000004
11.4 Precision: 0.18181818181818182
11.5 Recall: 0.03773584905660377
11.6 True Positive Rate: 0.03773584905660377
11.7 False Positive Rate: 0.061224489795918366
11.8 Specificity (TNR): 0.9387755102040817
11.9 F1 Score: 0.0625
```

11. Cross-Validation

```
cv_scores = cross_val_score(RandomForestClassifier(), X_scaled, y, cv=5)
print("12. Cross-validation scores:", cv_scores)
print("Mean CV Accuracy:", np.mean(cv_scores))
```

```
12. Cross-validation scores: [0.7   0.69  0.69  0.695 0.695]
Mean CV Accuracy: 0.694
```

12. Results and Conclusion

```
print("\n13. Results & Conclusion:")
best_model = max(accuracy_scores, key=accuracy_scores.get)
print(f"Best model: {best_model} with accuracy: {accuracy_scores[best_model]:.2f}")
```

```

13. Results & Conclusion:
Best model: Logistic Regression with accuracy: 0.73
```

13. References

```
print("\n14. References:")
print("- Scikit-learn documentation: https://scikit-learn.org/")
print("- Seaborn and Matplotlib for visualization")
print("- Data simulated for demo purposes")
```