Diabetes Prediction & Clustering

Index

Project Overview

The source of our project is a diabetes dataset from the National Institute of Diabetes and Digestive and Kidney Diseases.

It contains information on 768 women over 21 years of age and of Pima Indian heritage.

The data looks at different diagnostic parameters and whether or not the individual has Diabetes. These parameters are:

- Pregnancies
- Glucose
- Blood Pressure
- Skin Thickness
- Insulin
- Body Mass Index (BMI)
- Pedigree Function
- Age
- Diabetes Outcome

Our goal is to use these parameters and build machine learning models that predict the likeliness of an individual having Diabetes. Specifically we will be testing a variety of both supervised and unsupervised models like decision trees, random forest, support vector machines, and K-means clustering.

Cleaning

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

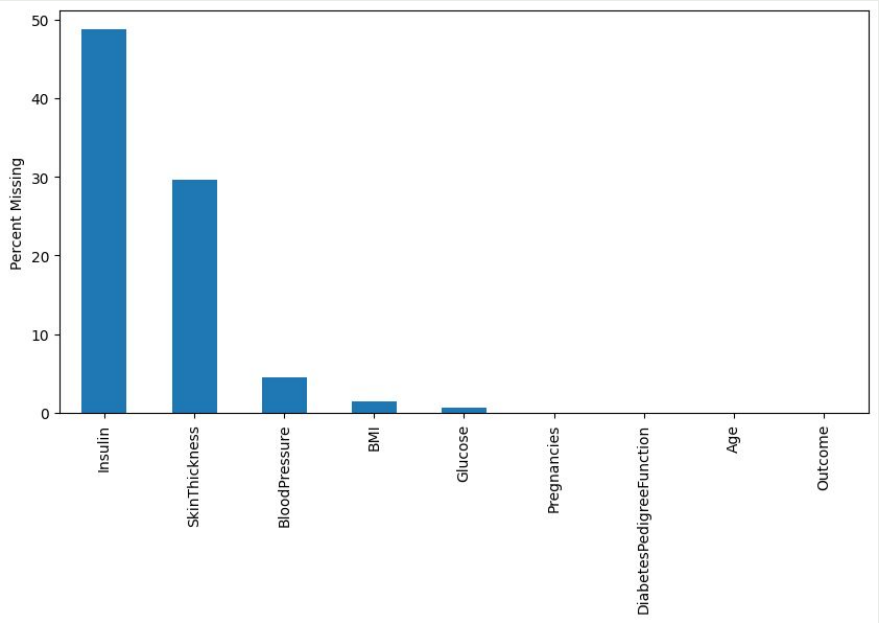
	0
Pregnancies	0
Glucose	0
BloodPressure	0
SkinThickness	0
Insulin	0
BMI	0
DiabetesPedigreeFunction	0
Age	0
Outcome	0

	0
Pregnancies	111
Glucose	5
BloodPressure	35
SkinThickness	227
Insulin	374
BMI	11
DiabetesPedigreeFunction	0
Age	0
Outcome	500

- Minimum values for Glucose, BloodPressure, SkinThickness, Insulin, and BMI are 0.
- Left shows total missing values
- Right shows total values that are 0.
- Missing values are being recorded as 0.

Cleaning

- Convert 0s into missing values for select columns.
- Graph proportion of missing values for each column
- Insulin and SkinThickness have a large amount of missing values
- Can’t just drop rows with missing values
- Drop columns or impute
- Continue with EDA



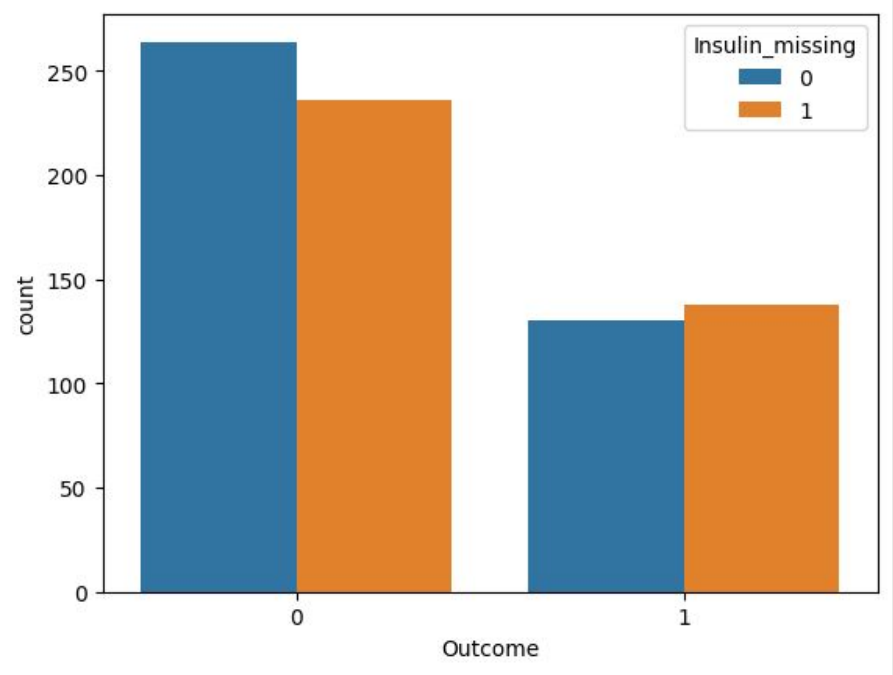
Exploratory Data Analysis

	Variable	Correlation	p-value	Significant (p<0.05)
0	Pregnancies	0.222	0.0	True
1	Glucose	0.495	0.0	True
2	BloodPressure	0.171	0.0	True
3	SkinThickness	0.259	0.0	True
4	Insulin	0.303	0.0	True
5	BMI	0.314	0.0	True
6	DiabetesPedigreeFunction	0.174	0.0	True
7	Age	0.238	0.0	True

- Run significance tests on all correlation coefficients with Outcome.
- All variables are significantly correlated with Outcome
- Insulin & SkinThickness despite high amounts of missing values also are significantly correlated.

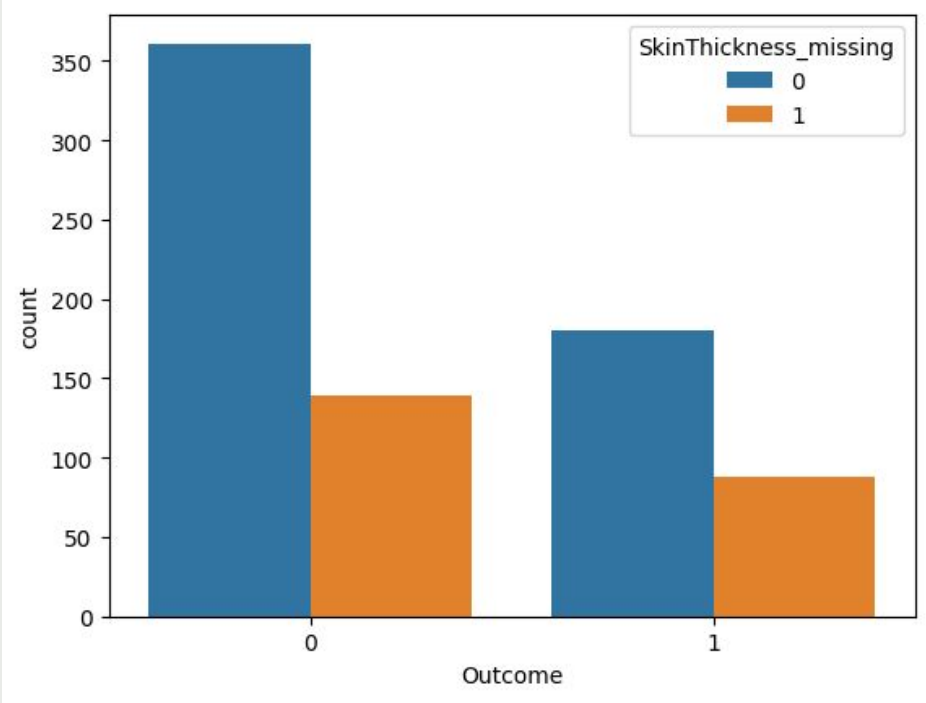
Exploratory Data Analysis

- Explore why Insulin and SkinThickness have so many missing values
- Graph of missing Insulin values for those with and without diabetes
- More missing values for those without diabetes.
- Suggests possible correlation
- Chi-squared test returns p-value of 0.2898
- 0.2898 > 0.05
- Missingness of Insulin is not significantly correlated with Outcome



Exploratory Data Analysis

- Graph of missing SkinThickness values for those with and without diabetes
- More missing values for those without diabetes
- Chi-squared test returns p-value of 0.1692
- 0.1692 > 0.05
- Missingness of SkinThickness is not significantly correlated with Outcome
- Missing indicators not needed for modeling
- Either impute values or drop columns.



Decision Tree

Methodology: Base models → Feature Importance Redefinition of Model → Tree Depth Optimization → Final Model

- Test Train 80/20 split, stratified
- Compared 2 methods of preprocessing:
  1. Drop Insulin
     - Accuracy: 0.792, F1: 0.704
  2. Median imputation
     - Accuracy: 0.792, F1: 0.704

```
# Model 1: Drop Insulin
X_train_ModelA = X_train.drop(columns = ['Insulin'])
X_test_ModelA = X_test.drop(columns = ['Insulin'])
dtree_ModelA = DecisionTreeClassifier(max_depth=3, random_state=0)
dtree_ModelA.fit(X_train_ModelA, y_train)
y_pred_ModelA = dtree_ModelA.predict(X_test_ModelA)
results["Drop Insulin"] = (accuracy_score(y_test, y_pred_ModelA), f1_score(y_test, y_pred_ModelA))


# Model 2: Impute Median
imputer = SimpleImputer(strategy='median')
X_train_ModelB = imputer.fit_transform(X_train)
X_test_ModelB = imputer.transform(X_test)
dtree_ModelB = DecisionTreeClassifier(max_depth=3, random_state=0)
dtree_ModelB.fit(X_train_ModelB, y_train)
y_pred_ModelB = dtree_ModelB.predict(X_test_ModelB)
results["Impute Median"] = (accuracy_score(y_test, y_pred_ModelB), f1_score(y_test, y_pred_ModelB))
```

Decision Tree

Optimization:

- Dropped 0 importance features
  - Zero-importance features: ['Pregnancies', 'BloodPressure', 'SkinThickness', 'Insulin']
- Compared tree depths (1-6) to maximize F1 score
  - Depth 4: F1 = 0.629

```
[30] #How much is Insulin affecting Decision Tree
     fi = pd.Series(dtree_ModelB.feature_importances_, index=X_train.columns)
     print(fi.sort_values(ascending=False))

     zero_features = fi[fi == 0].index.tolist()
     print("\nZero-importance features:", zero_features)
```

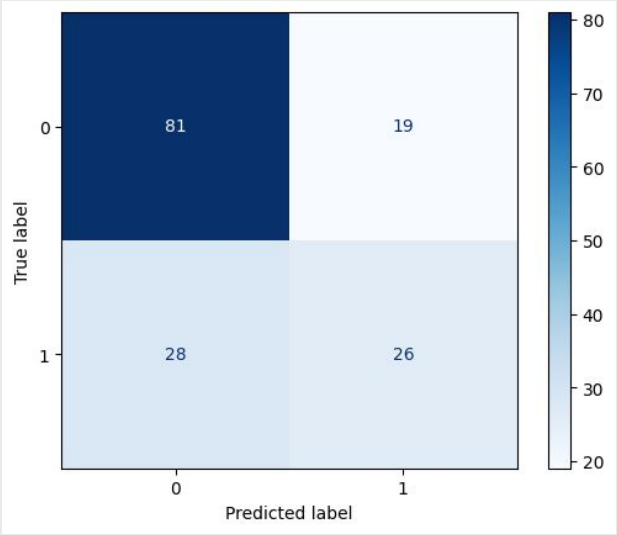
```
Glucose                     0.535321
BMI                         0.268310
Age                         0.138885
DiabetesPedigreeFunction    0.057485
SkinThickness               0.000000
BloodPressure               0.000000
Pregnancies                 0.000000
Insulin                     0.000000
```

```
Depth 1: F1 = 0.581
Depth 2: F1 = 0.572
Depth 3: F1 = 0.590
Depth 4: F1 = 0.629
Depth 5: F1 = 0.613
Depth 6: F1 = 0.578

Best depth: 4
Best F1: 0.629
```

Decision Tree

Results:

Model	Accuracy	F1:
Optimized Decision Tree	0.695	0.525



Takeaways:

- The Decision Tree does okay at identifying class 0 (non-Diabetic) but struggles more with class 1 (Diabetic):
- Low Recall which is crucial for our application

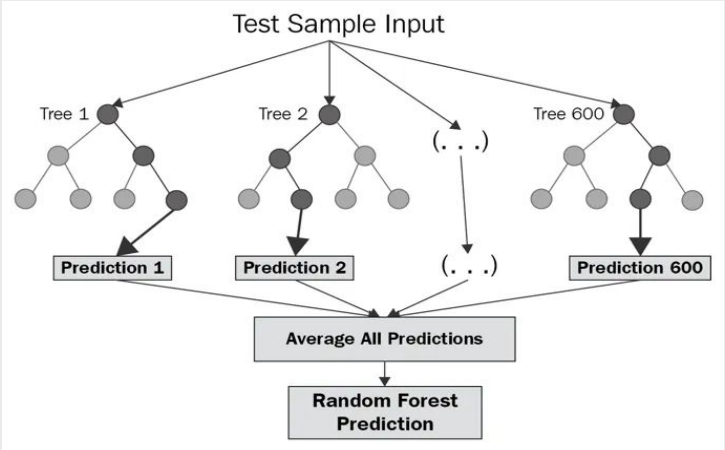
Random Forest

Methodology: Base model → Hyperparameter Tuning → Probability Threshold Tuning → Final Model

Base Model:

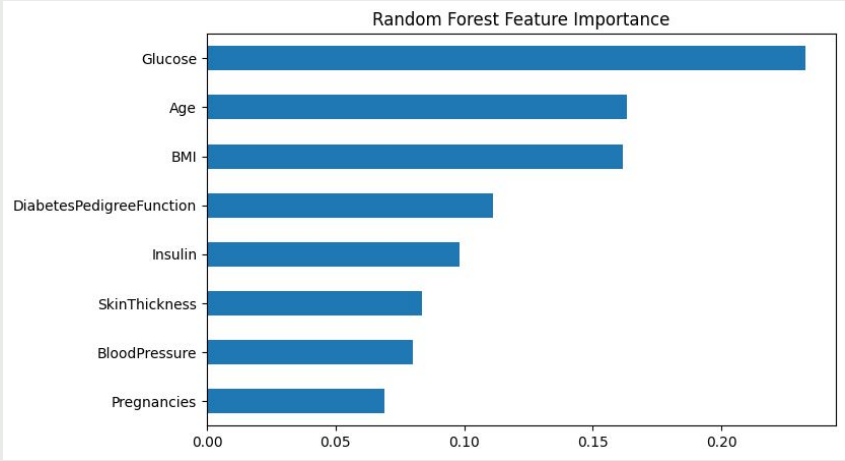
- Ensemble of decision trees
- Baseline Model n= 300
  - Model Accuracy: 0.81
  - F1-score : 0.694

```python
# Baseline Random Forest Model

# Initialize and train the Random Forest Classifier
rf_model = RandomForestClassifier(random_state=0, n_estimators=300)
rf_model.fit(X_train, y_train)

# Make prediction
y_pred = rf_model.predict(X_test)

# Evaluate the model's accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy:.2f}")

# F1-score
print(f"F1-score : {f1_score(y_test, y_pred):.3f}")
```



Random Forest

- **Hyperparameter tuning with GridSearchCV:**
  - n_estimators: 100, 300, 500
  - max_depth: None, 10, 20
  - max_features: sqrt, log2
  - class_weight: None, balanced
- Optimal parameters:




```
Best params: {'class_weight': 'balanced', 'max_depth': 10, 'max_features': 'sqrt', 'n_estimators': 100}
CV F1: 0.642
Accuracy: 0.792
Precision: 0.739
Recall: 0.63
F1: 0.642
```

Random Forest

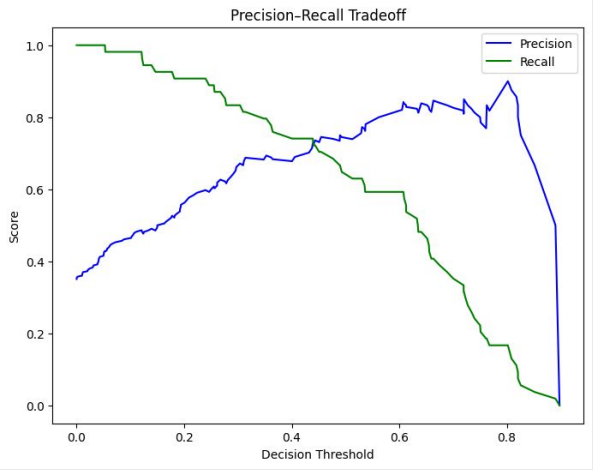
Threshold Probability Tuning:

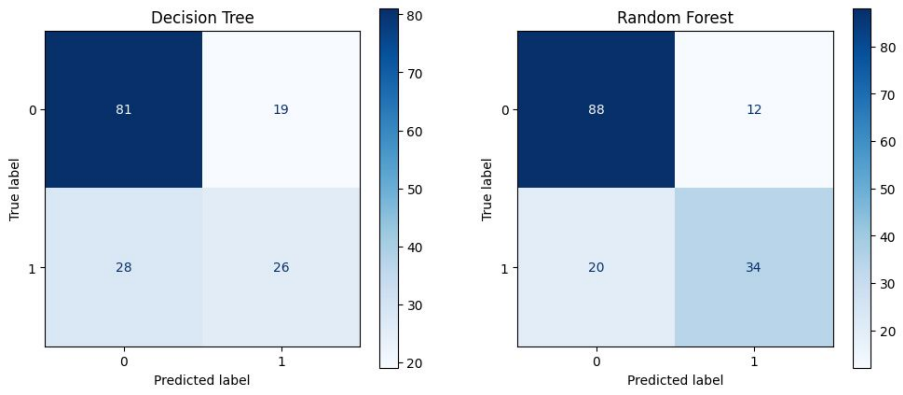
- Selecting probability threshold to improve F1 score
- Precision Recall Tradeoff Graph
  - Prioritizing Recall for Medical Diagnostic Application
- Best threshold for F1: 0.313

Final Optimized RF model Results:

- Accuracy: 0.805
- Precision: 0.688
- Recall: 0.815
- F1: 0.746

Takeaway: RF outperforms decision tree in all areas. Improved classification of Class 1 (Diabetes) and recall.





Support Vector Machines

Methodology

- Data cleaning → replace zeros with NaN, fill missing values with medians
- 80% training/ 20% test split with stratification to preserve class balance
- Standardization of features since SVMs are scale-sensitive
- Pipeline connects preprocessing + model for consistency and prevents leakage

Support Vector Machines

Hyper Parameter Tuning

- Used GridSearchCV to find the best combination of:
  - **C** (penalty strength)
  - **Kernel** (linear, polynomial, RBF)
  - **Gamma** (influence of data points for RBF/poly kernels)

- 5-fold cross-validation used to avoid overfitting.

- Correctly identified the kernel and parameters that produced the best accuracy.

- Rather than guessing, systematic tuning makes sure the model generalizes properly.

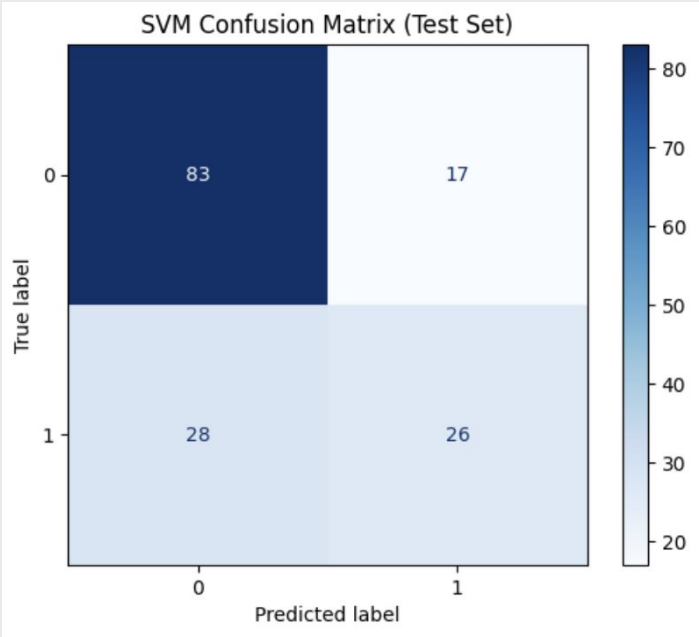
```
[172] # param combinations for tests
      param_grid = {
          "svm__C": [0.1, 1, 10],
          "svm__kernel": ["linear", "rbf", "poly"],
          "svm__gamma": ["scale", "auto"]
      }

      # cross-validation for each combination
      grid = GridSearchCV(pipe, param_grid, cv=5, scoring="accuracy", n_jobs=-1)
      grid.fit(X_train, y_train)

      print("Best params:", grid.best_params_)
      print("Best CV accuracy:", grid.best_score_)

      Best params: {'svm__C': 0.1, 'svm__gamma': 'scale', 'svm__kernel': 'linear'}
      Best CV accuracy: 0.775303212048514
```

Support Vector Machines

Confusion Matrix

- Decent performance, 71% accuracy → most patients correctly classified
- Strong performance on non-diabetic patients
- Weak performance on diabetic patients (48%). Further tuning, more features, or balancing techniques are needed to improve the model

	Precision	Recall	F1	Support
0	0.75	0.83	0.79	100
1	0.60	0.48	0.54	54
Accuracy			0.71	154
Macro Avg	0.68	0.66	0.66	154
Weighted Avg	0.70	0.71	0.70	154



Support Vector Machines

Model Comparison

- Decision Tree: weakest overall at 68% accuracy)
- Random Forest: strongest model at 81% accuracy, highest F1
- SVM: solid middle ground at 74% accuracy

- **Overall:** Random Forest provides the best balance of precision and recall
  - SVM can improve over a single Decision Tree but struggles with diabetic recall

	Accuracy	Precision	Recall	F1
Decision Tree	0.682	0.556	0.463	0.505
Random Forest	0.805	0.688	0.815	0.746
SVM	0.740	0.652	0.556	0.600

Support Vector Machines

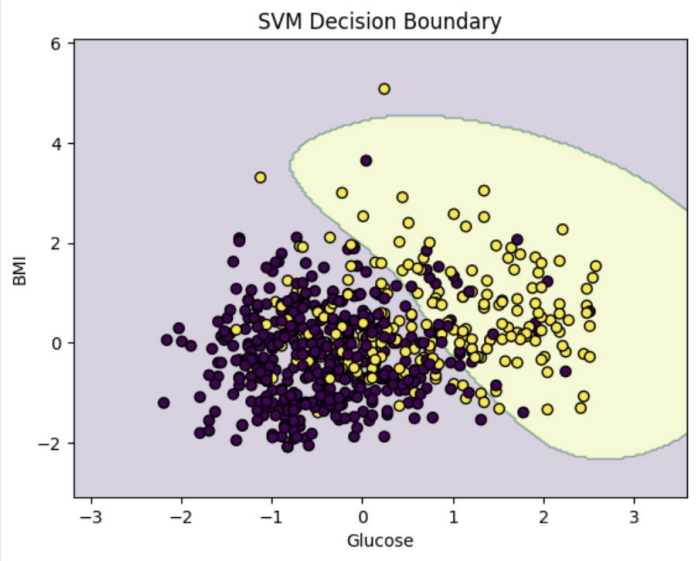
Decision Boundary

- Glucose and BMI features used for plotting
  - Diabetic = yellow
  - Non-diabetic = purple

- After scaling, plotted SVM's 2D decision boundary:
  - Shaded regions: model's predicted classes
  - Scatter points: actual patient outcomes

- Many diabetic patients falsely fall into the non-diabetic region

- SVM improves over other models, but the overlap shows why predicting diabetes is challenging with these features alone



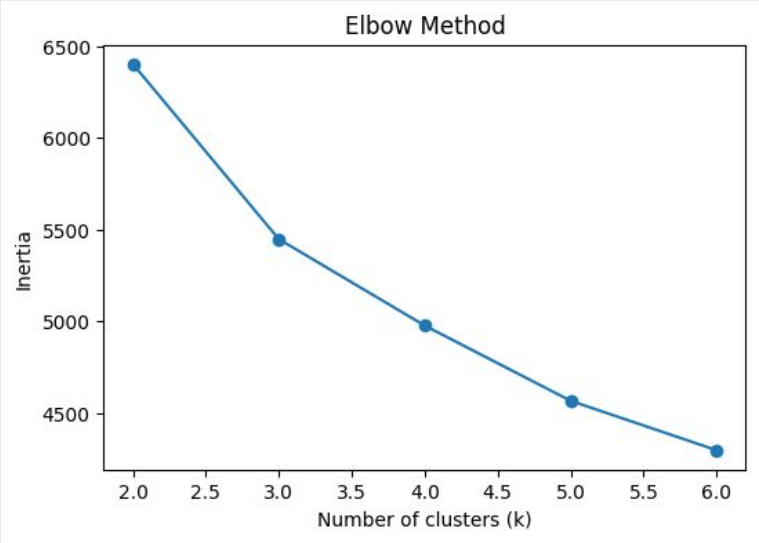
K-Means Clustering

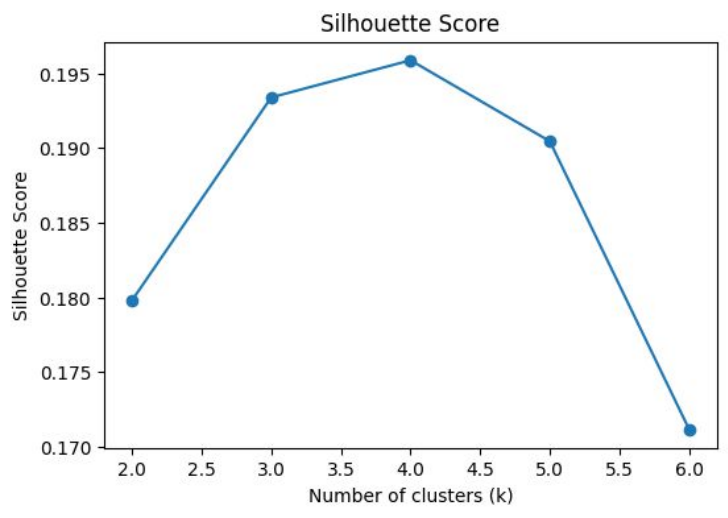
Preprocessing

- Dropped Outcome (labels) before clustering
- Use **median imputation** to replace NaN values
- Standardize features using **StandardScaler**

K-Means Clustering

Choosing Our k Value

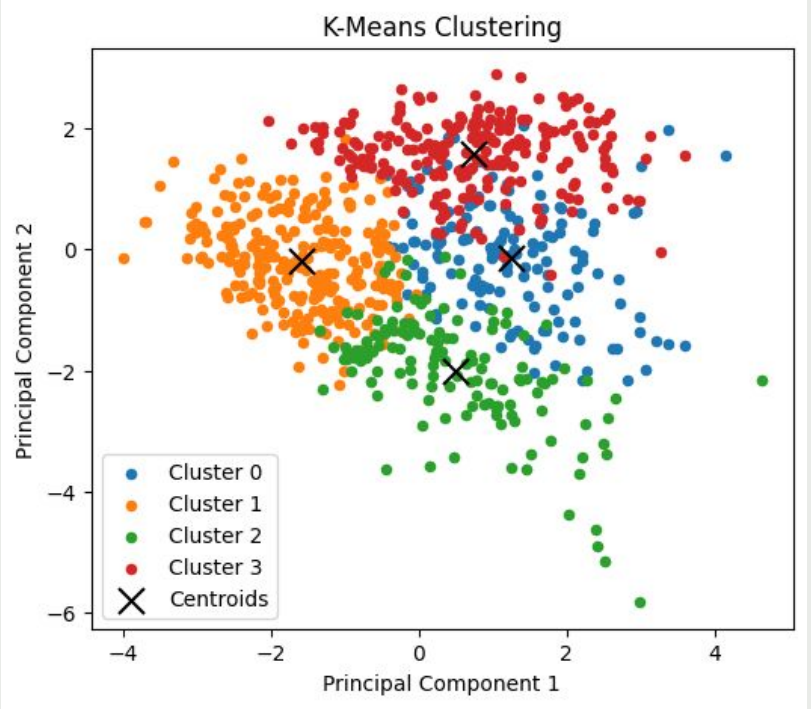




K-Means Clustering

Results

- 4 clusters shown in PCA 2D projection
- Shows **natural separation** of patients into groups



K-Means Clustering

Evaluation vs Outcome

- Purity: ~67%
- Adjusted Rand Index (ARI): 0.078
- Captures some diabetes-related grouping, but also highlights other hidden subgroups

Cluster	Outcome 0	Outcome 1
0	67	80
1	225	35
2	70	67
3	138	86

Thank You!