DIABETES PREDICTION

ASHISH JOSHI

Date : 26-03-2025

Introduction

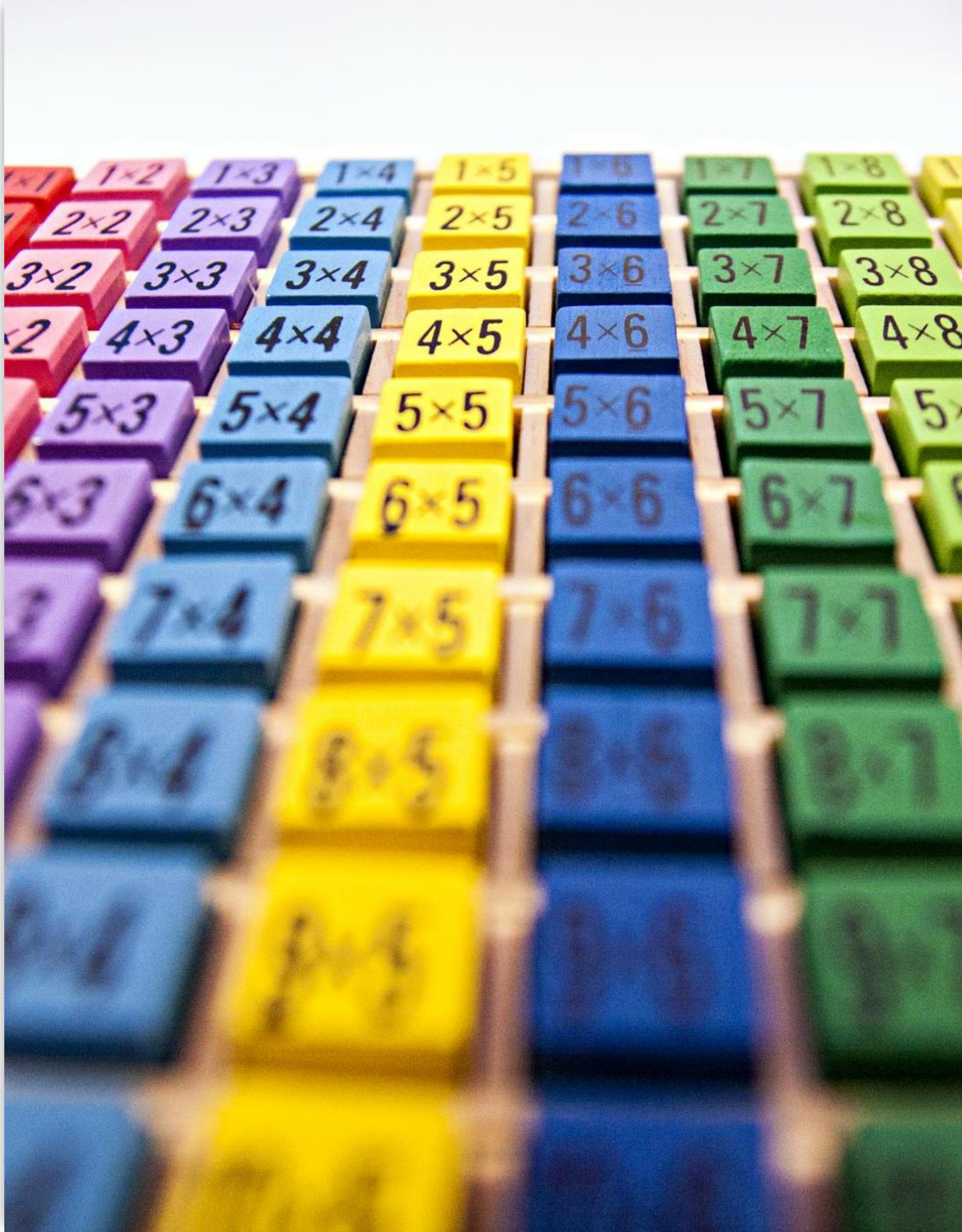
- **What is Diabetes?**
  A disease where the body cannot manage blood sugar properly.
- **Why Predict Diabetes?**
  Early prediction helps prevent complications and improves health.
- **Goal of the Project:**
  To predict diabetes using machine learning models for accurate results

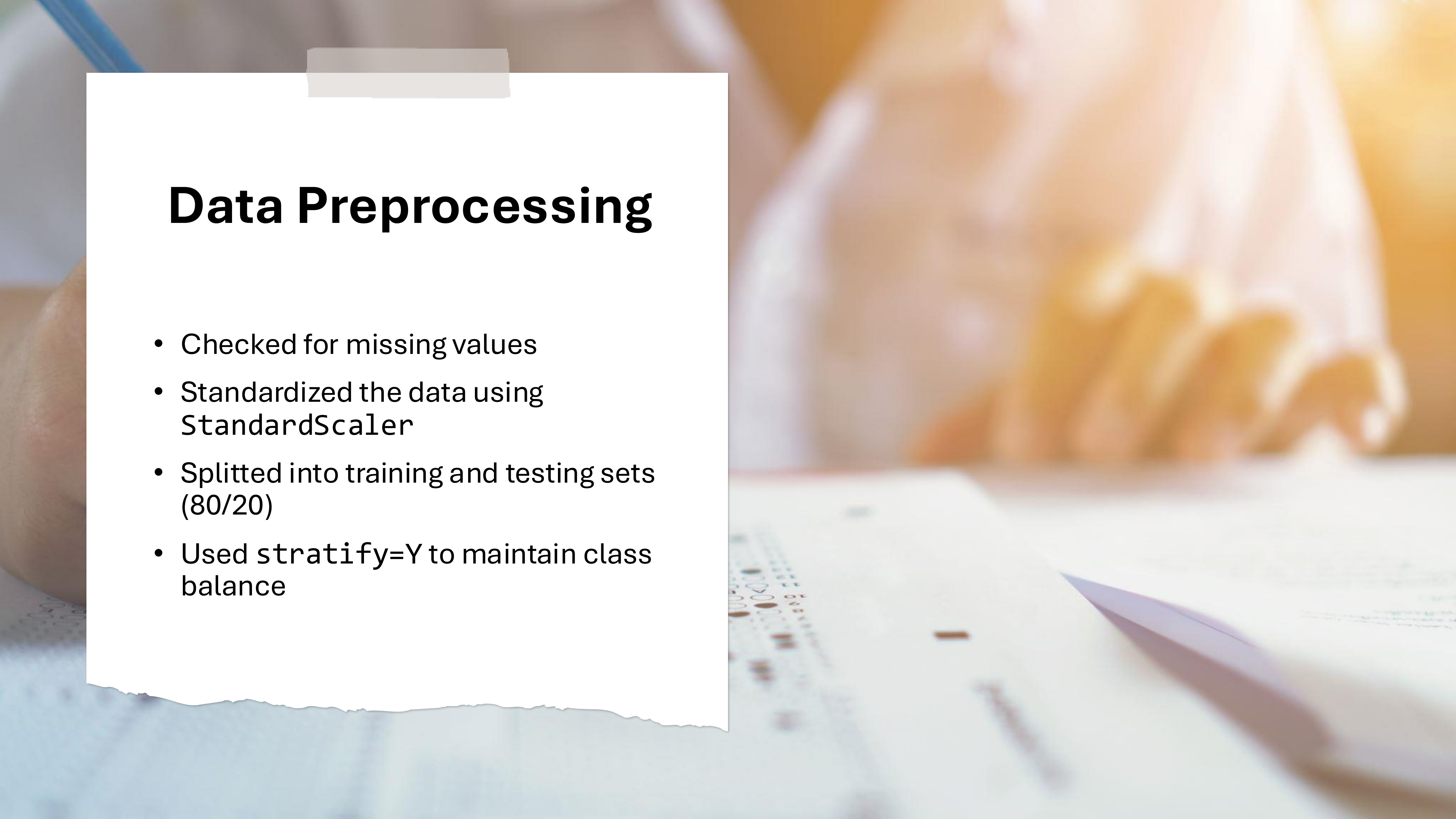
Objective

- The objective of this project is to develop a machine learning model that predicts whether a person is diabetic based on health parameters such as age, BMI, glucose levels, and blood pressure. The goal is to create an accurate system that can assist in early diabetes detection for better prevention and management.

Dataset Description

- **Dataset**: Pima Indian Diabetes Dataset (from Kaggle/UCI)
- **Rows**: 768
- **Features**: 8 input features + 1 output label
    - Examples of features: Glucose, BMI, Age, Insulin, etc.
- **Output label:**
- 1: Diabetic
- 0: Non-diabetic



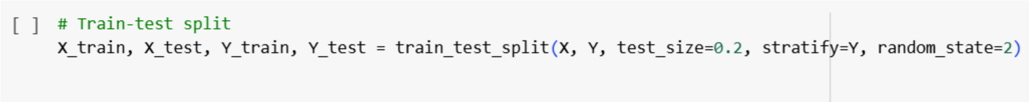


Data Preprocessing

- Checked for missing values
- Standardized the data using `StandardScaler`
- Splitted into training and testing sets (80/20)
- Used `stratify=Y` to maintain class balance

Train-Test Split

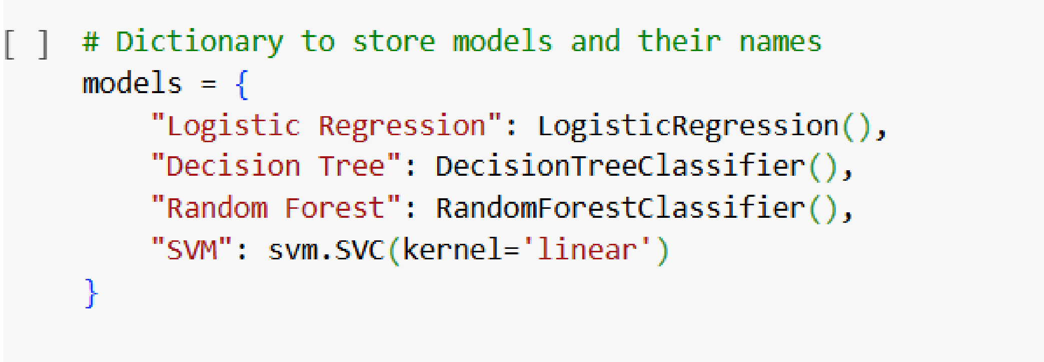
- Splits dataset into:
  - **80% training**
  - **20% testing**
- Stratified so both sets have similar class balance (important for fairness).



```
# Train-test split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)
```

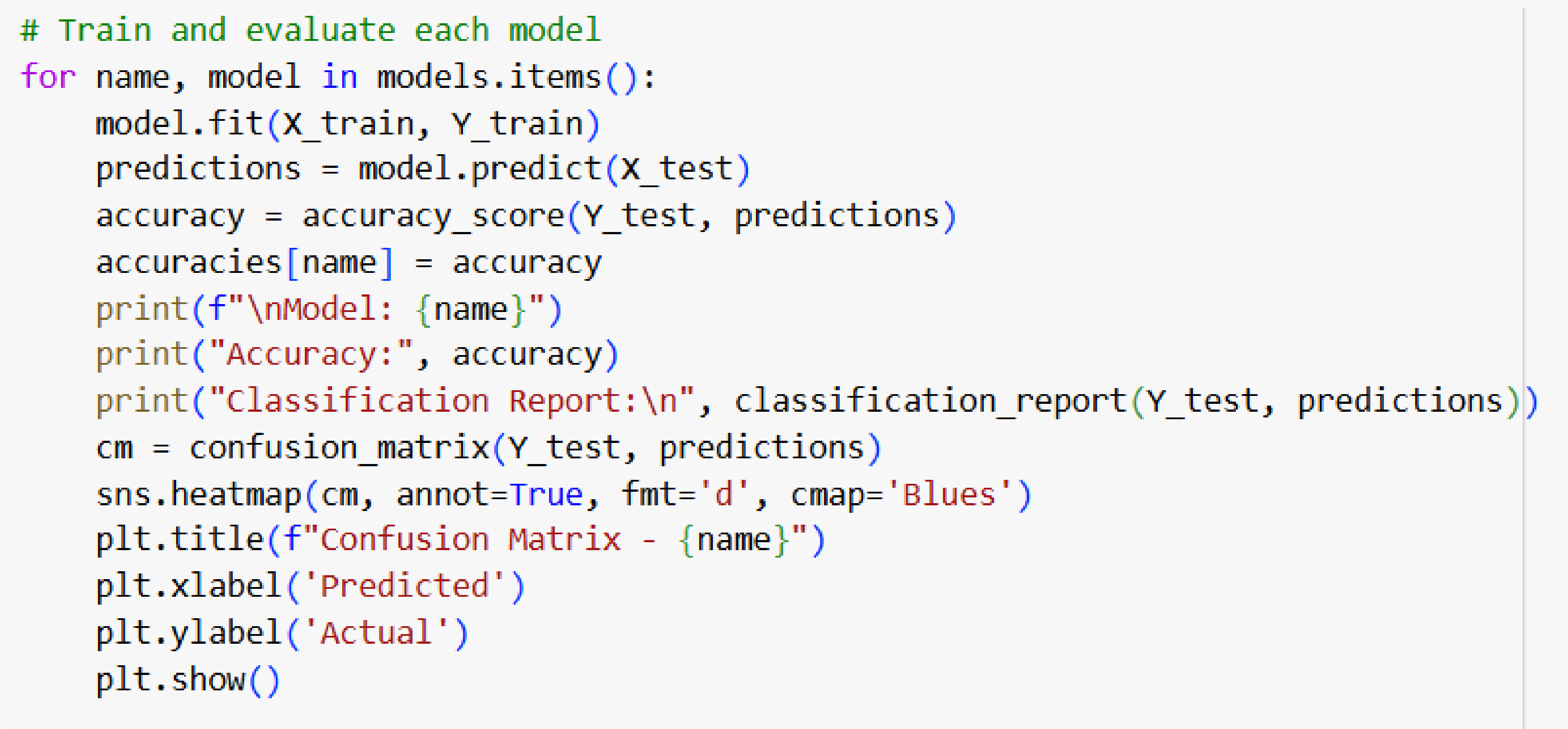
6. **Model Selection & Training**

- 4 model
- Logistic Regression
- Decision Tree
- Random Forest
- Support Vector Machine (SVM)



```
# Dictionary to store models and their names
models = {
    "Logistic Regression": LogisticRegression(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "SVM": svm.SVC(kernel='linear')
}
```

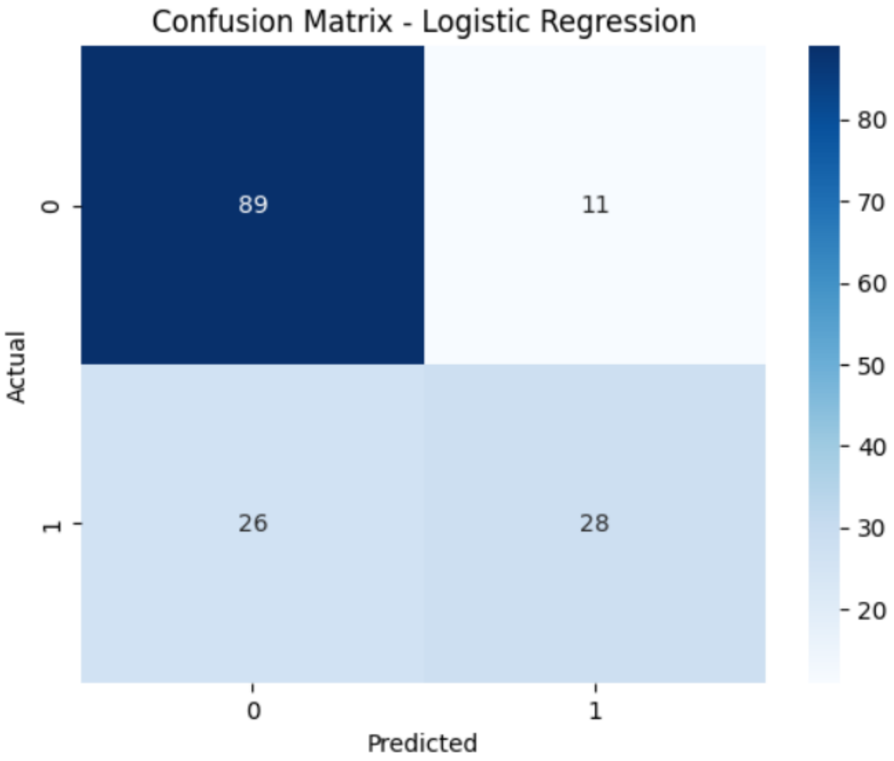
Each model is trained using:



```
# Train and evaluate each model
for name, model in models.items():
    model.fit(X_train, Y_train)
    predictions = model.predict(X_test)
    accuracy = accuracy_score(Y_test, predictions)
    accuracies[name] = accuracy
    print(f"\nModel: {name}")
    print("Accuracy:", accuracy)
    print("Classification Report:\n", classification_report(Y_test, predictions))
    cm = confusion_matrix(Y_test, predictions)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title(f"Confusion Matrix - {name}")
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()
```

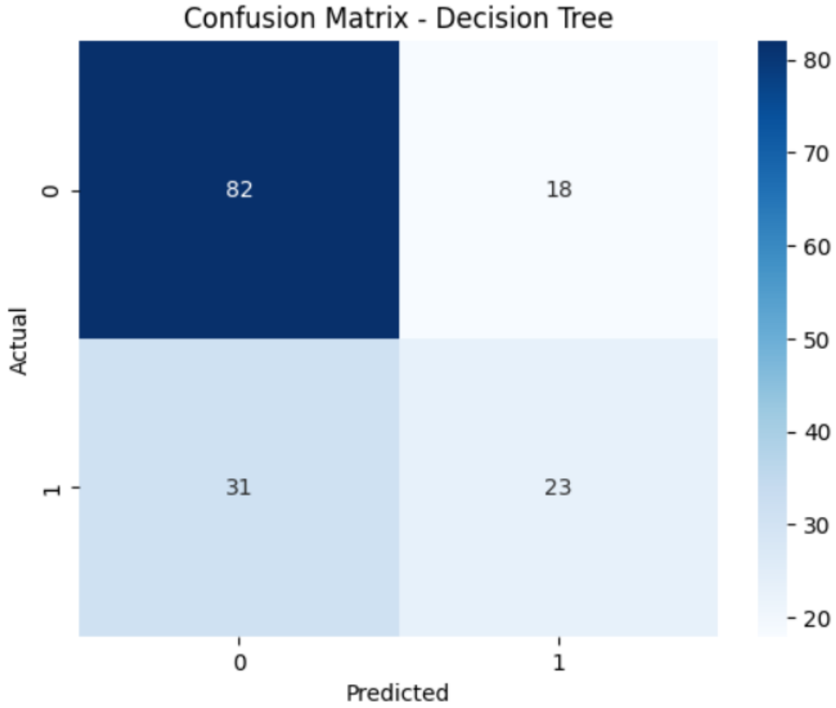
Model 1 Logistic Regression

- **Accuracy Score:** 75.97%
- **Classification Report Highlights:**
  - **Precision:** 0.77 (Class 0), 0.72 (Class 1)
  - **Recall:** 0.89 (Class 0), 0.52 (Class 1)
  - **F1-Score:** 0.83 (Class 0), 0.60 (Class 1)
  - **Macro Avg:** Precision: 0.75, Recall: 0.70, F1-Score: 0.72
  - **Weighted Avg:** Precision: 0.75, Recall: 0.76, F1-Score: 0.75
- **Confusion Matrix:**



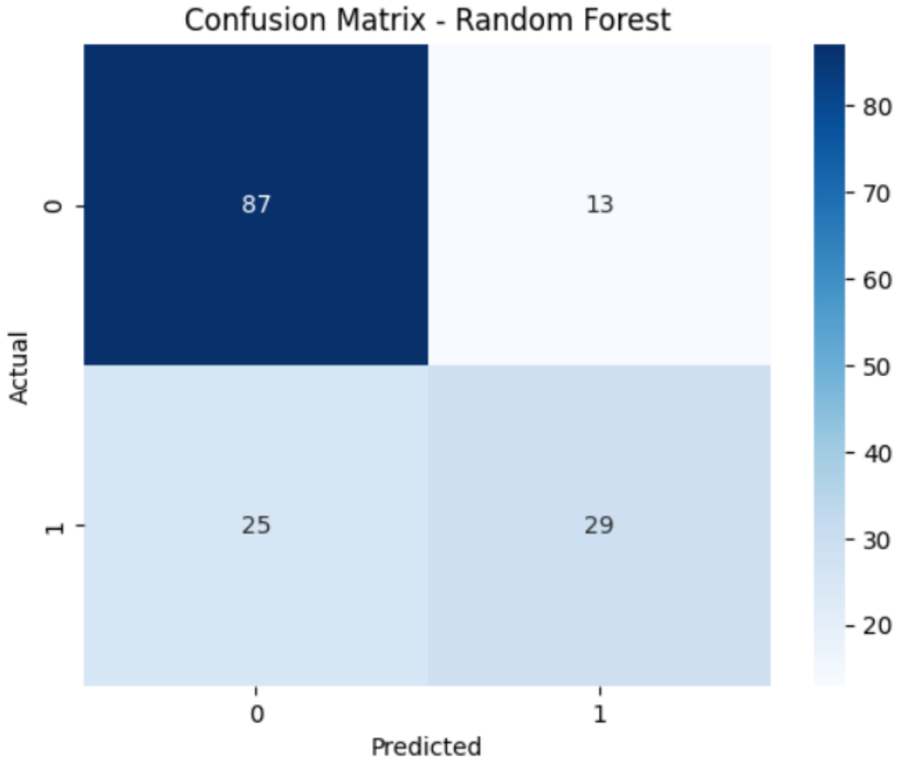
Modal 2 Decision Tree

- **Accuracy Score:** 68.18%
- **Classification Report Highlights:**
  - **Precision:** 0.73 (Class 0), 0.56 (Class 1)
  - **Recall:** 0.82 (Class 0), 0.43 (Class 1)
  - **F1-Score:** 0.77 (Class 0), 0.48 (Class 1)
  - **Macro Avg:** Precision: 0.64, Recall: 0.62, F1-Score: 0.63
  - **Weighted Avg:** Precision: 0.67, Recall: 0.68, F1-Score: 0.67
- **Confusion Matrix:**



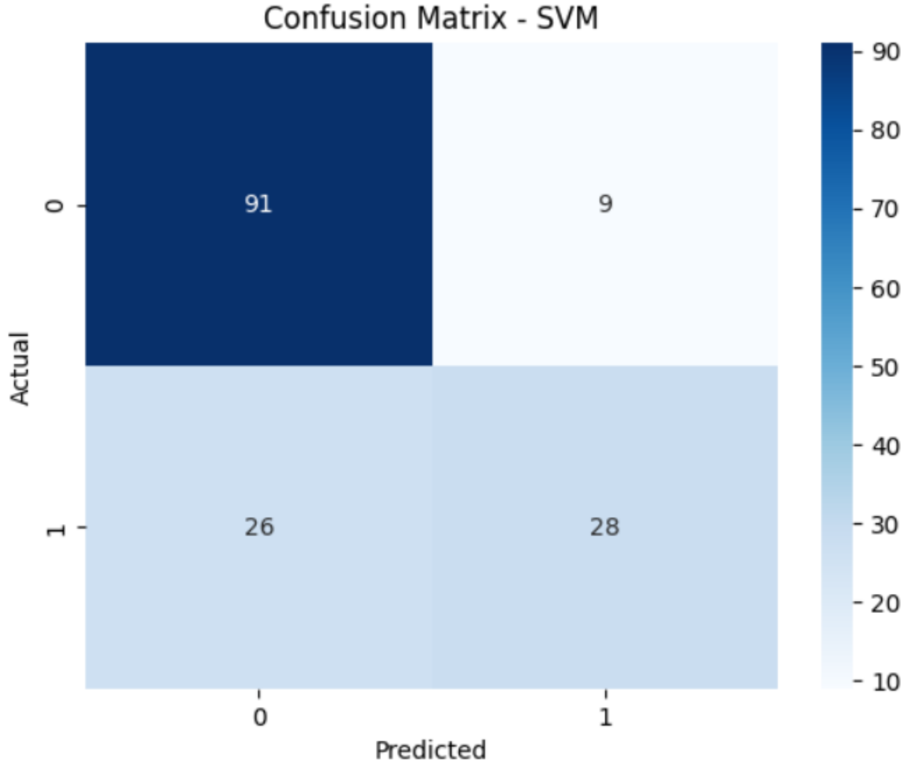
• **Model 3:** Random Forest

- **Accuracy Score:** 75.32%
- **Classification Report Highlights:**
  - **Precision:** 0.78 (Class 0), 0.69 (Class 1)
  - **Recall:** 0.87 (Class 0), 0.54 (Class 1)
  - **F1-Score:** 0.82 (Class 0), 0.60 (Class 1)
  - **Macro Avg:** Precision: 0.73, Recall: 0.70, F1-Score: 0.71
  - **Weighted Avg:** Precision: 0.75, Recall: 0.75, F1-Score: 0.74
- **Confusion Matrix:**



Model 4: Support Vector Machine (SVM)

- **Model Name:** Support Vector Machine (SVM)
- **Accuracy Score:** 77.27%
- **Classification Report Highlights:**
  - **Precision:** 0.78 (Class 0), 0.76 (Class 1)
  - **Recall:** 0.91 (Class 0), 0.52 (Class 1)
  - **F1-Score:** 0.84 (Class 0), 0.62 (Class 1)
  - **Macro Avg:** Precision: 0.77, Recall: 0.71, F1-Score: 0.73
  - **Weighted Avg:** Precision: 0.77, Recall: 0.77, F1-Score: 0.76
- **Confusion Matrix:**

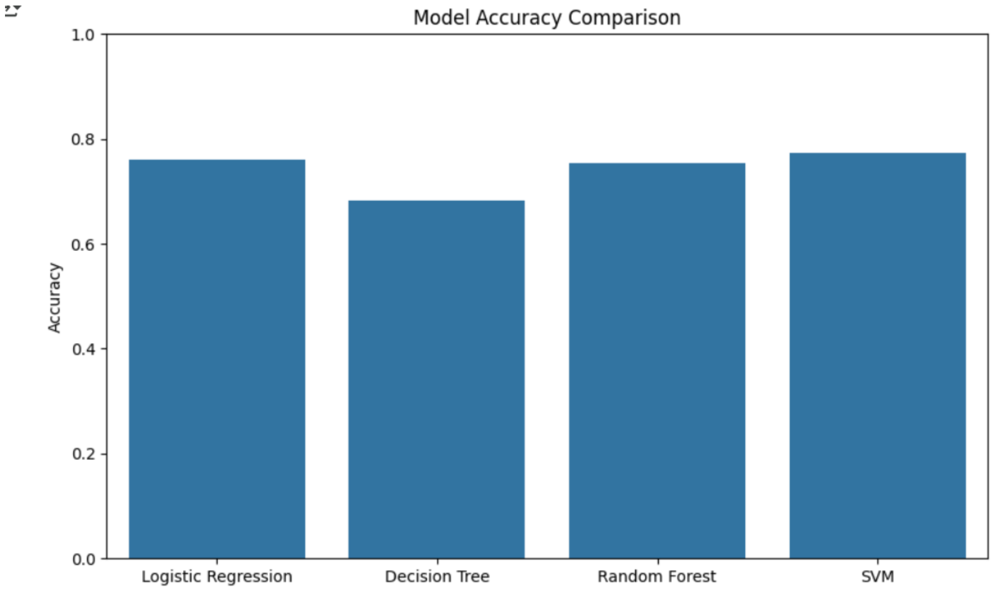


Accuracy Comparison

Bar Chart:

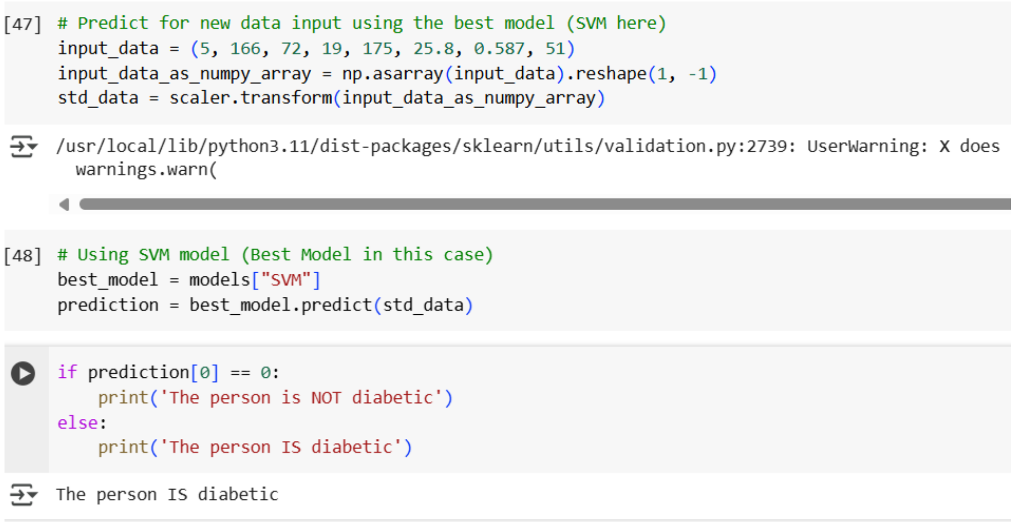
- Display a bar chart comparing the accuracy of the 4 models:
  - Logistic Regression: 75.97%
  - Decision Tree: 68.18%
  - Random Forest: 75.32%
  - SVM: 77.27%

- **Highlight the Best Performing Model:**
  - **Support Vector Machine (SVM)** with an accuracy of **77.27%** is the best performing model.



Final Prediction Example

- **Input Data:**
  (5, 166, 72, 19, 175, 25.8, 0.587, 51)
  (Pregnancies = 5, Glucose = 166, Blood Pressure = 72, Skin Thickness = 19, Insulin = 175, BMI = 25.8, Diabetes Pedigree Function = 0.587, Age = 51)
- **Model Used:** SVM
- **Prediction:**
  - **The person IS diabetic**



```
[47] # Predict for new data input using the best model (SVM here)
     input_data = (5, 166, 72, 19, 175, 25.8, 0.587, 51)
     input_data_as_numpy_array = np.asarray(input_data).reshape(1, -1)
     std_data = scaler.transform(input_data_as_numpy_array)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does
  warnings.warn(
```

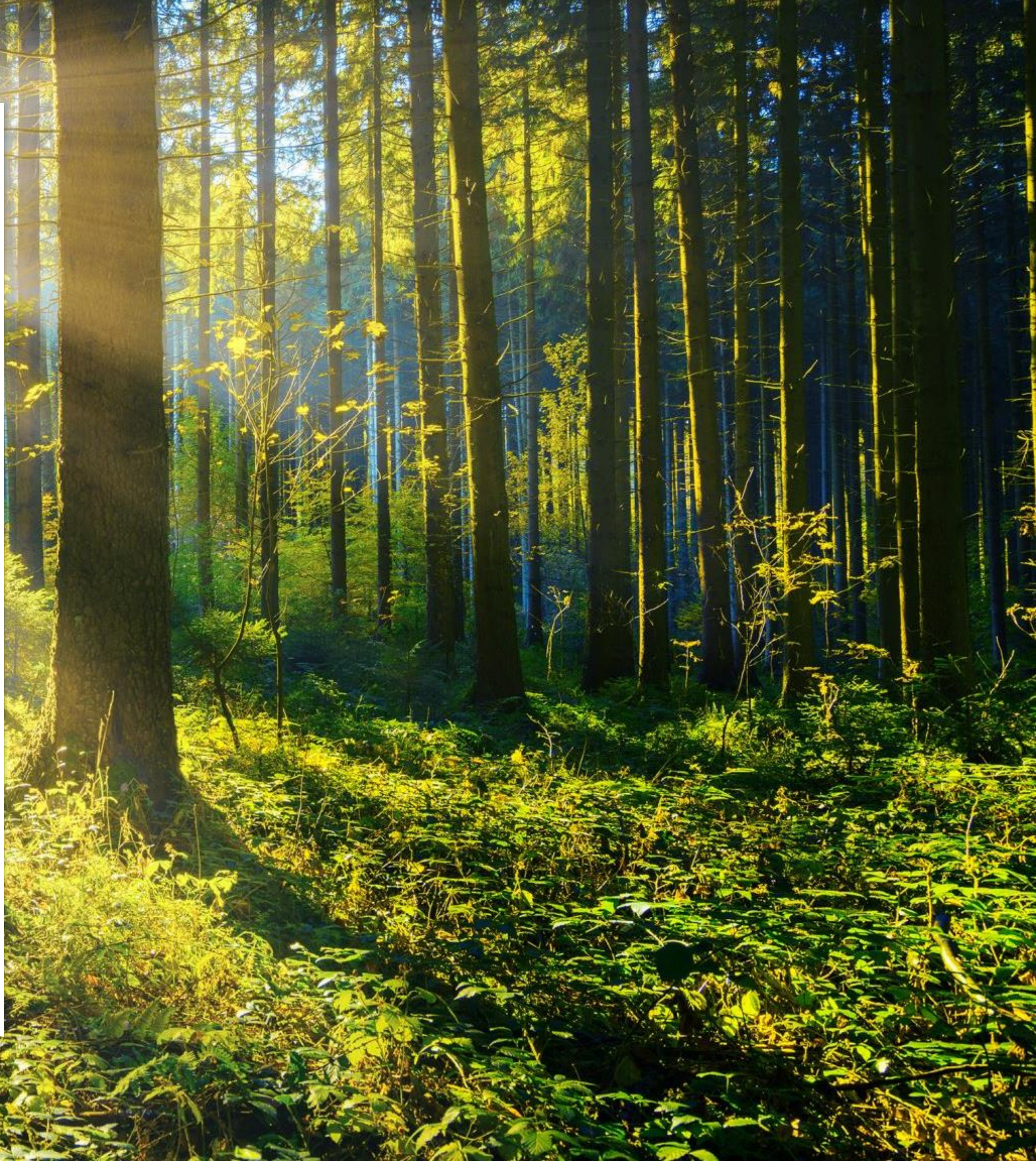
```
[48] # Using SVM model (Best Model in this case)
     best_model = models["SVM"]
     prediction = best_model.predict(std_data)
```

```
if prediction[0] == 0:
    print('The person is NOT diabetic')
else:
    print('The person IS diabetic')
```

```
The person IS diabetic
```

Conclusion

- SVM gave the highest accuracy
- Machine learning can be a useful tool in healthcare
- This system can help in early detection and management of diabetes



Future Improvements

1. Add more models (like KNN, XGBoost)
2. Include ROC-AUC and cross-validation
3. Deploy as a **web app using Streamlit**
4. Use a larger or real-time dataset

"Thank you!"
“Any questions?”