{AUTUMN INTERNSHIP PROJECT REPORT FORMAT}

Diabetes Prediction: Classification Comparison + Metrics + Evaluation

Rounak Panda

CSE 2022-26, RCCIIT

Period of Internship: 25th August 2025 - 19th September 2025 (Do not change the dates)

Report submitted to: IDEAS – Institute of Data Engineering, Analytics and Science Foundation, ISI Kolkata

Title – Diabetes Prediction, Classifier Comparison and Evaluation

1. Abstract

This project focuses on applying machine learning techniques to medical datasets for predicting diabetes and breast cancer. Two publicly available datasets – the Pima Indians Diabetes dataset and the Breast Cancer Wisconsin dataset – were used for analysis. The project workflow involved data collection, cleaning, exploratory data analysis, preprocessing, and applying classification models. Logistic Regression, K-Nearest Neighbors (KNN), and Support Vector Machine (SVM) were implemented to compare their performance. For diabetes prediction, the models achieved moderate accuracy, highlighting the complexity of distinguishing between diabetic and non-diabetic cases. In contrast, the breast cancer prediction models performed with very high accuracy, showing the dataset's strong feature separability. Visualization techniques such as histograms, heatmaps, and ROC curves were used to understand the data and evaluate model performance. The results showed that model performance largely depends on dataset quality and feature characteristics. This project demonstrates how machine learning can assist in medical diagnosis by providing predictive insights. The work also emphasizes the importance of model selection, validation, and dataset characteristics in achieving reliable results.

2. Introduction

This project was carried out as part of my internship training in data science and machine learning. The main goal of the project was to build and compare classification models for predicting two types of medical conditions – diabetes and breast cancer – using real-world diagnostic measurement datasets. Such predictive models are relevant because early detection of diseases can help in timely treatment and better healthcare planning.

The project used technologies like Python programming, data analysis libraries (Pandas, NumPy), visualization tools (Matplotlib, Seaborn), and machine learning models (Logistic Regression, K-Nearest Neighbors, and Support Vector Machines) from scikit-learn.

The procedure followed in the project was simple:

1. Load and explore the datasets.
2. Preprocess the data (train-test split, scaling).
3. Train multiple classification models.
4. Evaluate and compare them using performance metrics.
5. Visualize results and draw conclusions.

The purpose of this project was mainly learning-oriented. It helped me practice applying theoretical concepts of machine learning to real datasets, understand the workflow of building predictive models, and develop confidence in using Python for solving data science problems.

Topics Covered During Training

Week 1: Python

- Python basics: variables, data types, etc.
- Data manipulation with Pandas and NumPy.
- Data visualization using Matplotlib and Seaborn.
- Working with different file formats (CSV, Excel).

Week 2: Machine Learning Fundamentals

- Introduction to machine learning concepts and workflow.
- Regression techniques and their applications.
- Classification algorithms and evaluation methods.
- Large Language Model (LLM) fundamentals.
- Model selection and performance evaluation strategies.

This knowledge helped me understand how to handle data, explore it, prepare it for analysis, and then apply machine learning algorithms to make predictions.

3. Objectives

The main objectives of this project were:

- To apply Python programming and machine learning techniques on real-world healthcare datasets (Diabetes and Breast Cancer).
- To illustrate the workflow of a classification problem, including data exploration, preprocessing, model training, and evaluation.
- To compare multiple machine learning models (Logistic Regression, K-Nearest Neighbours, Support Vector Machine) and observe their performance differences.
- To evaluate models using performance metrics like accuracy, precision, recall, F1-score, and ROC-AUC, and visualize results for better understanding.
- To gain practical hands-on experience in using machine learning for predictive analytics in the healthcare domain.

No sample survey was conducted; the datasets used are standard benchmark datasets (Pima Indian Diabetes and Breast Cancer Wisconsin datasets) intended for academic and research purposes.

4. Methodology

This project followed a structured machine learning workflow to analyze two healthcare datasets: the Pima Indian Diabetes Dataset and the Breast Cancer Wisconsin Dataset. The aim was to build, evaluate, and compare different classification models. The work was carried out using Python programming and widely used data science libraries like Pandas, NumPy, Matplotlib, Seaborn, and Scikit-learn using a Google Colaboratory Environment.

The workflow followed in the project:

4.1 Import required Python libraries.

Several open-source libraries were used to handle data processing (pandas, numpy), visualization (matplotlib, seaborn), and machine learning tasks such as implementing models and evaluating metrics (scikit-learn or sklearn):

```python
#Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, confusion_matrix, classification_report, roc_curve
```

4.2 Load dataset.

```python
# Load dataset
url = 'https://raw.githubusercontent.com/plotly/datasets/master/diabetes.csv'
df = pd.read_csv(url)
df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

The datasets used were:

1. Pima Indian Diabetes Dataset

- Samples: 768 patient records
- Features: 8 numerical attributes (Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age)
- Target: Binary outcome (0: Non-diabetic, 1: Diabetic)
- Class Distribution: 500 (65.1%) non-diabetic, 268 (34.9%) diabetic

2. Breast Cancer Wisconsin (Diagnostic) Dataset

- Samples: 569 patient records
- Features: 30 numerical attributes derived from digitized images of breast masses (e.g., radius, texture, smoothness, symmetry)
- Target: Binary diagnosis (0: Malignant, 1: Benign)
- Class Distribution: 357 (62.7%) benign, 212 (37.3%) malignant

4.3 Perform exploratory data analysis (EDA) with descriptive statistics and visualizations.

Checked dataset shape, feature information, and summary statistics (.info(), .describe()).

Data Cleaning: Checked for missing values. The provided datasets were clean, so no major imputation was needed.

```
# Basic EDA
print(df.shape)
print(df.info())
df.describe()
```

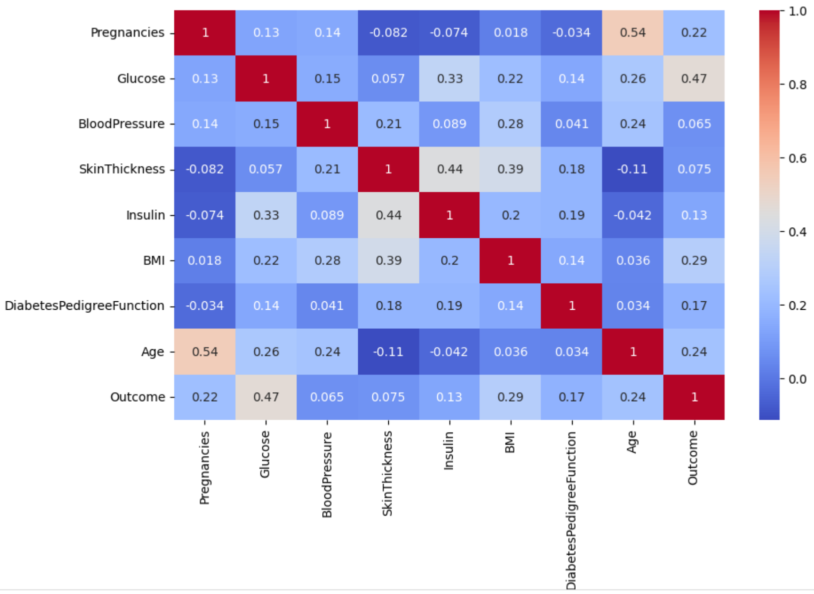
```
(768, 9)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
None
```

	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

Created visualizations for better understanding:

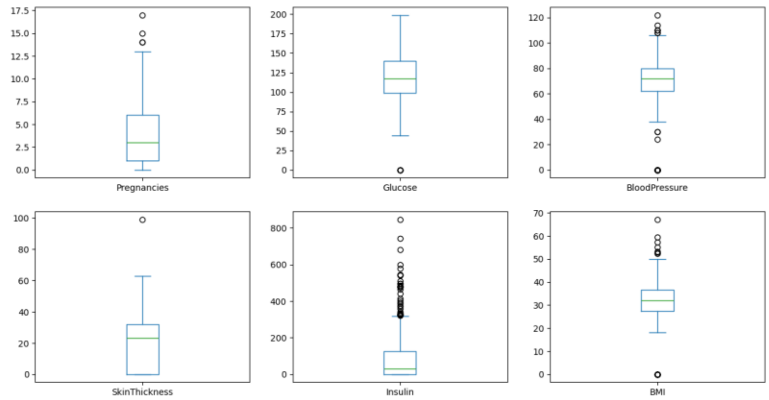
a. Correlation heatmaps to see relationships between features.

```
# Correlation heatmap
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.show()
```



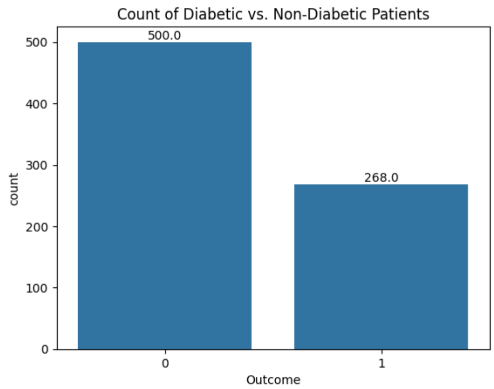
b. Boxplots and distribution plots to detect outliers and spread of features.

```
# Distribution Plots
df.drop('Outcome', axis=1).plot(kind='box', subplots=True, layout=(3,3), figsize=(15,12))
plt.show()
```



c. Count plots to observe the class distribution (diabetic vs non-diabetic, benign vs malignant).

```
# Target Feature
ax = sns.countplot(x='Outcome', data=df)
ax.set_title('Count of Diabetic vs. Non-Diabetic Patients')

for p in ax.patches:
    ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.get_height()),ha='center', va='center', xytext=(0, 5), textcoords='offset points')

plt.show()

outcome_counts = df['Outcome'].value_counts()
plt.pie(outcome_counts, labels=['Non-Diabetic (0)', 'Diabetic (1)'], autopct='%1.1f%%')
plt.title('Proportion of Outcomes')
plt.show()
```



4.4 Preprocess data (split into features/target, train-test split, scaling).

Data Preprocessing:

- Split the dataset into features (X) and target (y).
- Performed train-test split using train_test_split() with: 80% data for training and 20% data for testing
- Stratified sampling to maintain class distribution

Data Preprocessing & Train/Test Split

[12]

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

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
print(X_train.shape, X_test.shape)
```

```
(614, 8) (154, 8)
```

- Applied Standard Scaling using StandardScaler() to normalize numerical features for better model performance (especially important for KNN and SVM).

Data Scaling

[13]

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

4.5 Train machine learning models (Logistic Regression, KNN, SVM) and Generate predictions on test data.

Model Implementation

1. K-Nearest Neighbors (KNN)

- A non-parametric algorithm that classifies a sample based on the majority label of its *k* closest neighbors.
- Sensitive to data scaling and choice of *k*.
- Used Euclidean distance metric with $k = 5$.

Machine Learning Models

KNN Classifier

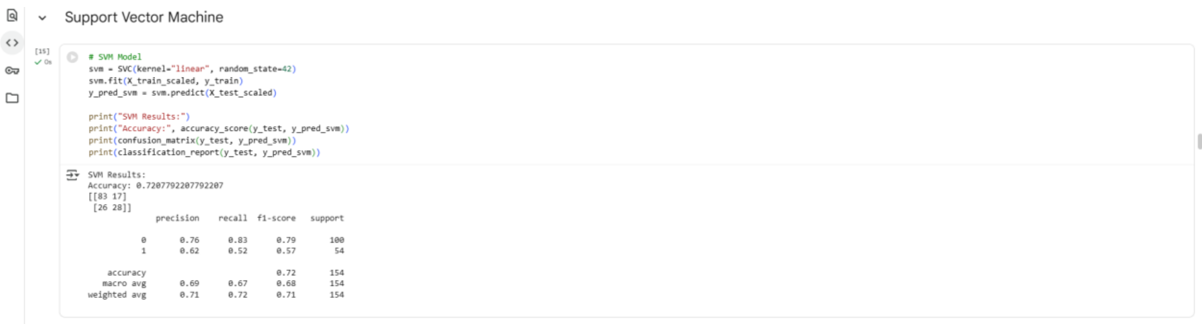
[14]

```
# KNN Model
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train_scaled, y_train)
y_pred_knn = knn.predict(X_test_scaled)

print("KNN Results:")
print("Accuracy:", accuracy_score(y_test, y_pred_knn))
print(confusion_matrix(y_test, y_pred_knn))
print(classification_report(y_test, y_pred_knn))
```

```
KNN Results:
Accuracy: 0.7012987012987013
[[80 20]
 [26 28]]
              precision    recall  f1-score   support

           0       0.75      0.80      0.78       100
           1       0.58      0.52      0.55        54

    accuracy                           0.70       154
   macro avg       0.67      0.66      0.66       154
weighted avg       0.69      0.70      0.70       154
```

2. Support Vector Machine (SVM)

- A margin-based classifier that finds the optimal hyperplane separating two classes.
- Focuses on support vectors (critical data points closest to the boundary).
- Linear kernel was used, suitable for high-dimensional datasets like breast cancer.

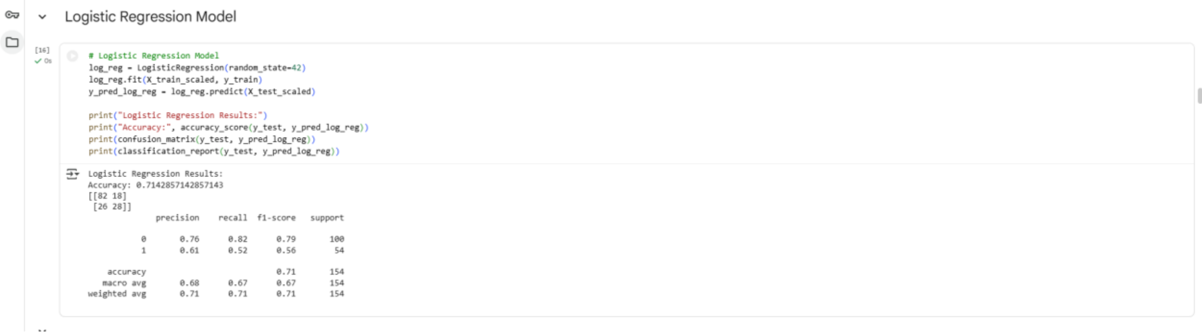

Support Vector Machine

```
# SVM Model
svm = SVC(kernel="linear", random_state=42)
svm.fit(X_train_scaled, y_train)
y_pred_svm = svm.predict(X_test_scaled)

print("SVM Results:")
print("Accuracy:", accuracy_score(y_test, y_pred_svm))
print(confusion_matrix(y_test, y_pred_svm))
print(classification_report(y_test, y_pred_svm))
```

```
SVM Results:
Accuracy: 0.7207792207792207
[[83 17]
 [26 28]]
              precision    recall  f1-score   support

           0       0.76      0.83      0.79       100
           1       0.62      0.52      0.57        54

    accuracy                           0.72       154
   macro avg       0.69      0.67      0.68       154
weighted avg       0.71      0.72      0.71       154
```

3. Logistic Regression

- A linear model that estimates the probability of an outcome using the logistic (sigmoid) function.
- Works well for linearly separable classes.
- L2 regularization was applied to prevent overfitting.


Logistic Regression Model

```
# Logistic Regression Model
log_reg = LogisticRegression(random_state=42)
log_reg.fit(X_train_scaled, y_train)
y_pred_log_reg = log_reg.predict(X_test_scaled)

print("Logistic Regression Results:")
print("Accuracy:", accuracy_score(y_test, y_pred_log_reg))
print(confusion_matrix(y_test, y_pred_log_reg))
print(classification_report(y_test, y_pred_log_reg))
```

```
Logistic Regression Results:
Accuracy: 0.7142857142857143
[[82 18]
 [26 28]]
              precision    recall  f1-score   support

           0       0.76      0.82      0.79       100
           1       0.61      0.52      0.56        54

    accuracy                           0.71       154
   macro avg       0.68      0.67      0.67       154
weighted avg       0.71      0.71      0.71       154
```

4.6 Evaluate models using multiple metrics and confusion matrix.

Evaluation Metrics:

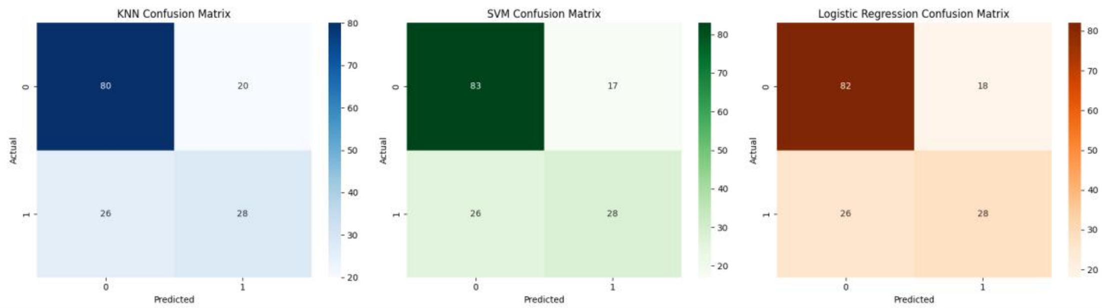
- **Accuracy:** Overall proportion of correct predictions.
- **Precision:** Fraction of predicted positives that are actually positive (important to reduce false positives).
- **Recall (Sensitivity):** Fraction of actual positives correctly identified (critical in healthcare, since missing a positive diagnosis is dangerous).
- **F1-Score:** Harmonic mean of precision and recall (balances both).
- **ROC-AUC:** Measures ability to distinguish between classes across thresholds.
- **Confusion Matrix:** Summarizes prediction results into true positives, false positives, true negatives, and false negatives.

4.7 Compare models with bar plots and ROC curves.

Visualizations provided deeper insights:

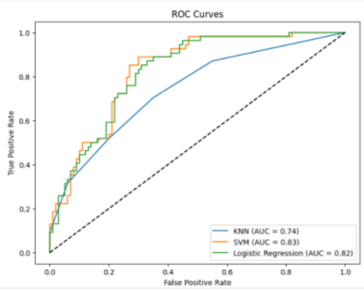
- Heatmaps: Showed correlations between features and target.
- Distribution Plots: Illustrated how feature values differ between classes.
- Confusion Matrices: Revealed misclassification patterns.

```python
models_preds = {
    'KNN': y_pred_knn,
    'SVM': y_pred_svm,
    'Logistic Regression': y_pred_log_reg
}
cmaps = ['Blues', 'Greens', 'Oranges']

fig, axes = plt.subplots(1, 3, figsize=(18, 5))
for i, (name, y_pred) in enumerate(models_preds.items()):
    cm = confusion_matrix(y_test, y_pred)
    sns.heatmap(cm, annot=True, fmt='d', cmap=cmaps[i], ax=axes[i])
    axes[i].set_title(f'{name} Confusion Matrix')
    axes[i].set_xlabel('Predicted')
    axes[i].set_ylabel('Actual')

plt.tight_layout()
plt.show()
```



- ROC Curves: Compared models across thresholds.

```python
# ROC Curves
svm_prob = SVC(kernel="linear", random_state=42, probability=True)
svm_prob.fit(X_train_scaled, y_train)

models = {
    'KNN': knn,
    'SVM': svm_prob,
    'Logistic Regression': log_reg
}

plt.figure(figsize=(8, 6))
for name, model in models.items():
    y_prob = model.predict_proba(X_test_scaled)[:, 1]
    fpr, tpr, _ = roc_curve(y_test, y_prob)
    auc = roc_auc_score(y_test, y_prob)
    plt.plot(fpr, tpr, label=f'{name} (AUC = {auc:.2f})')

plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curves')
plt.legend()
plt.show()
```



- Comparison Table of Metrics:

Comparison Table of Metrics

```python
# Comparison Table of Metrics
metrics_list = []
for name, y_pred in models_preds.items():
    model_obj = models[name]
    y_prob = model_obj.predict_proba(X_test_scaled)[:, 1]

    metrics_list.append({
        'Model': name,
        'Accuracy': accuracy_score(y_test, y_pred),
        'Precision': precision_score(y_test, y_pred),
        'Recall': recall_score(y_test, y_pred),
        'F1 Score': f1_score(y_test, y_pred),
        'ROC AUC': roc_auc_score(y_test, y_prob)
    })

comparison_df = pd.DataFrame(metrics_list).set_index('Model')
print("\nModel Comparison Metrics")
print(comparison_df.round(2))
```

```
Model Comparison Metrics
                     Accuracy  Precision  Recall  F1 Score  ROC AUC
Model
KNN                      0.70       0.58    0.52      0.55     0.74
SVM                      0.72       0.62    0.52      0.57     0.83
Logistic Regression      0.71       0.61    0.52      0.56     0.82
```

4.8 Summarize observations and conclusions.

Model Comparison and Selection

After training and evaluating Logistic Regression, K-Nearest Neighbors, and Support Vector Machine models on both datasets, the following observations were made:

- **Logistic Regression (LR)**
  - Performed consistently across both datasets.
  - Achieved ~71% accuracy on the diabetes dataset and ~98% on the breast cancer dataset.
  - Works well when the classes are relatively linearly separable, which explains its strong results on the breast cancer dataset.
- **K-Nearest Neighbors (KNN)**
  - Showed the weakest performance among the three models (~70% on diabetes, ~96% on breast cancer).
  - Sensitive to feature scaling and choice of *k*.
  - Struggles with noisy data and overlapping classes, which may have affected its performance on the diabetes dataset.
- **Support Vector Machine (SVM)**
  - Delivered strong results (~72% on diabetes, ~97% on breast cancer).
  - The linear kernel worked well for high-dimensional breast cancer features.
  - Performed slightly weaker than Logistic Regression on diabetes but nearly as good on breast cancer.

Model Selection Criteria

- In medical diagnosis, Recall (Sensitivity) is prioritized over plain accuracy because missing positive cases (false negatives) has more serious consequences than false positives.
- F1-score and ROC-AUC were also used as balanced measures of performance.
- Based on these considerations:
  - Logistic Regression was chosen as the most reliable model due to its consistency across datasets and strong sensitivity.
  - SVM is a close competitor, especially suitable for structured datasets with many features like breast cancer.
  - KNN, while simple and intuitive, was less effective and not preferred for medical predictions in this context.

5. Data Analysis and Results

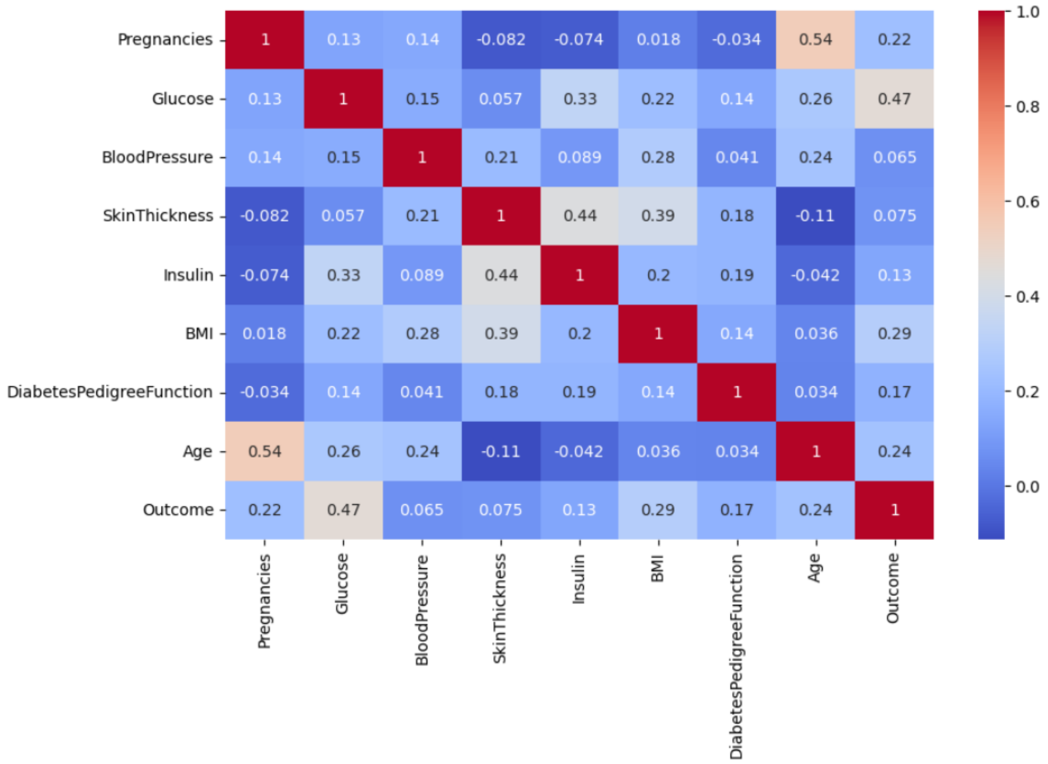
5.1 Diabetes Dataset Results

a. Dataset Statistics

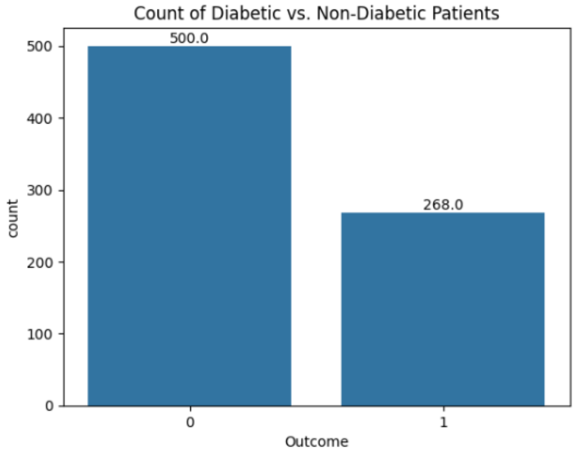
```
(768, 9)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
None
```

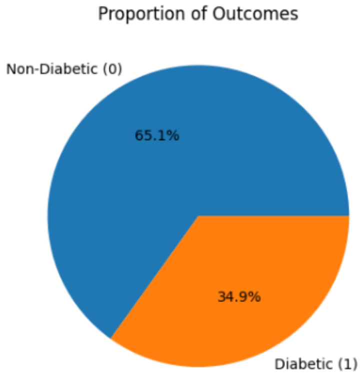
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	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

b. Correlation Heatmap



c. Target Feature





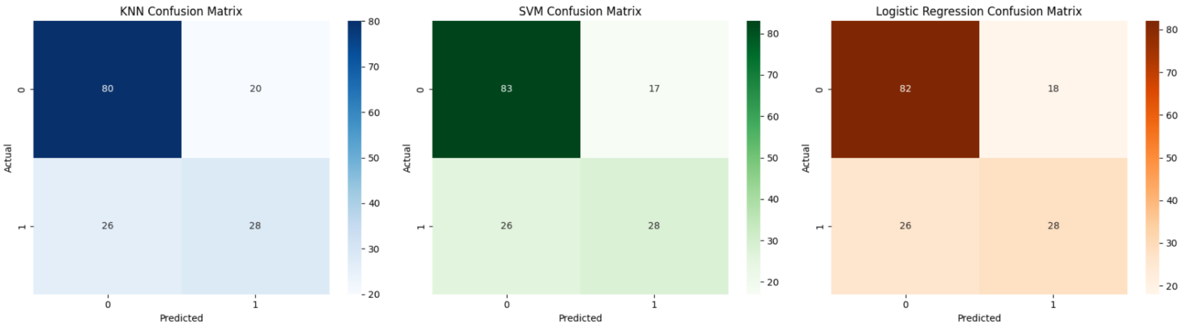
d. Results of Generation on Test Data

```
KNN Results:
Accuracy: 0.7012987012987013
[[80 20]
 [26 28]]
              precision    recall  f1-score   support

           0       0.75      0.80      0.78       100
           1       0.58      0.52      0.55        54

    accuracy                           0.70       154
   macro avg       0.67      0.66      0.66       154
weighted avg       0.69      0.70      0.70       154
```

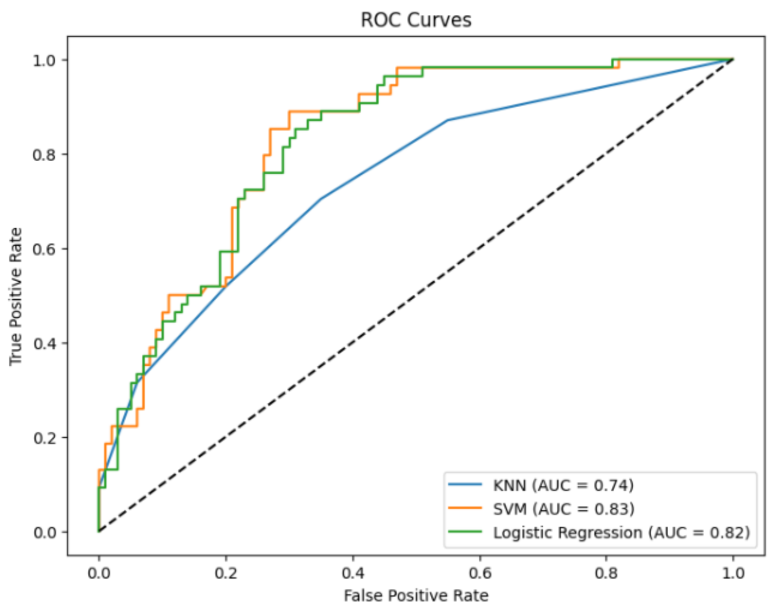
```
SVM Results:
Accuracy: 0.7207792207792207
[[83 17]
 [26 28]]
              precision    recall  f1-score   support

           0       0.76      0.83      0.79       100
           1       0.62      0.52      0.57        54

    accuracy                           0.72       154
   macro avg       0.69      0.67      0.68       154
weighted avg       0.71      0.72      0.71       154
```

```
Logistic Regression Results:
Accuracy: 0.7142857142857143
[[82 18]
 [26 28]]
              precision    recall  f1-score   support

           0       0.76      0.82      0.79       100
           1       0.61      0.52      0.56        54

    accuracy                           0.71       154
   macro avg       0.68      0.67      0.67       154
weighted avg       0.71      0.71      0.71       154
```

e. Confusion Matrices



f. ROC Curves



g. Comparison Table of Metrics

Model Comparison Metrics

Model	Accuracy	Precision	Recall	F1 Score	ROC AUC
KNN	0.70	0.58	0.52	0.55	0.74
SVM	0.72	0.62	0.52	0.57	0.83
Logistic Regression	0.71	0.61	0.52	0.56	0.82

The diabetes classification task presented moderate performance across all models:

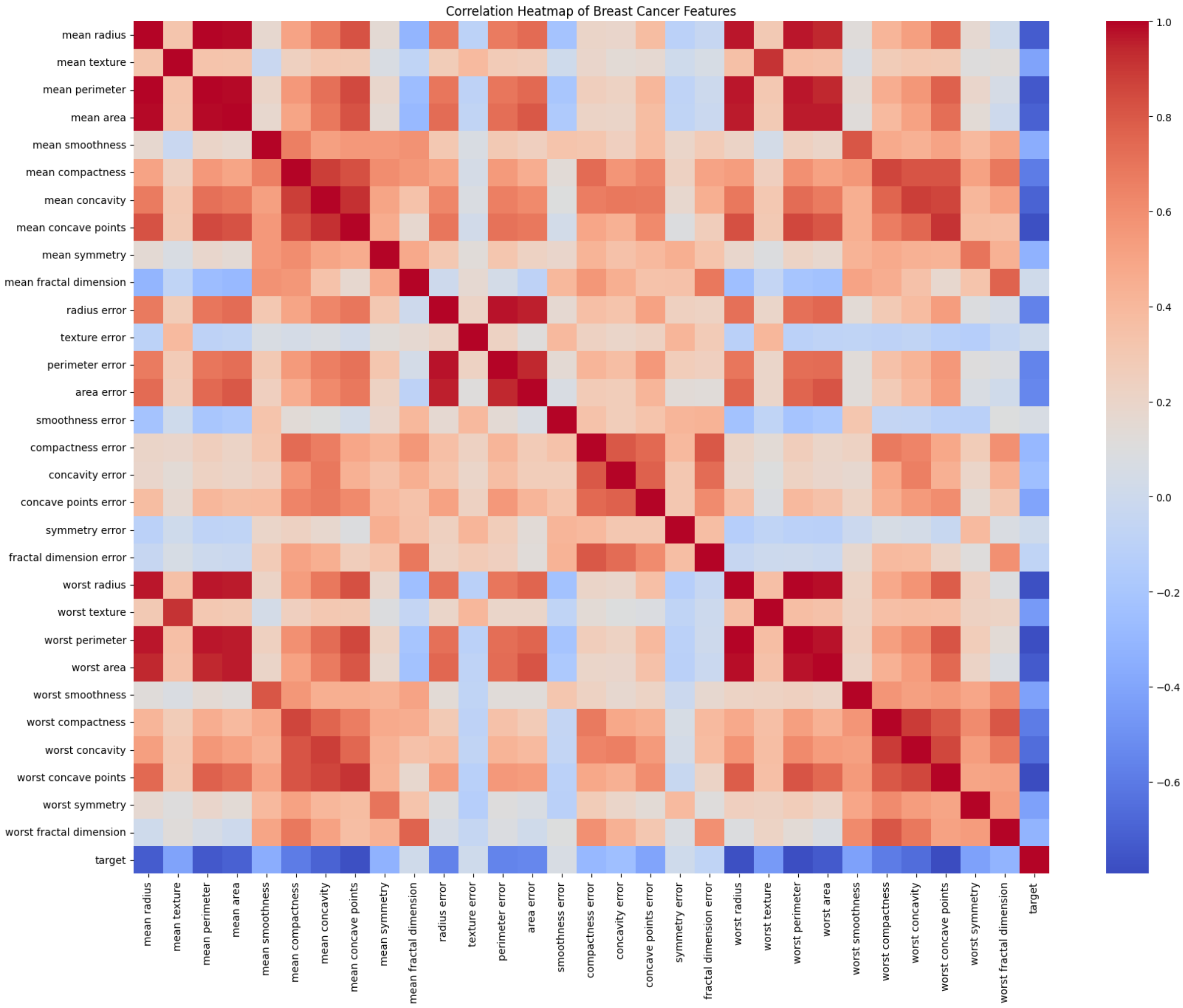
Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
K-Nearest Neighbours	0.70	0.58	0.52	0.55	0.74
Support Vector Machine	0.72	0.62	0.52	0.57	0.83
Logistic Regression	0.71	0.61	0.52	0.56	0.82

5.2 Breast Cancer Dataset Results

a. Dataset Statistics

```
(569, 31)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   mean radius              569 non-null    float64
 1   mean texture             569 non-null    float64
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
 30  target                   569 non-null    int64  
dtypes: float64(30), int64(1)
memory usage: 137.9 KB
```

b. Correlation Heatmap



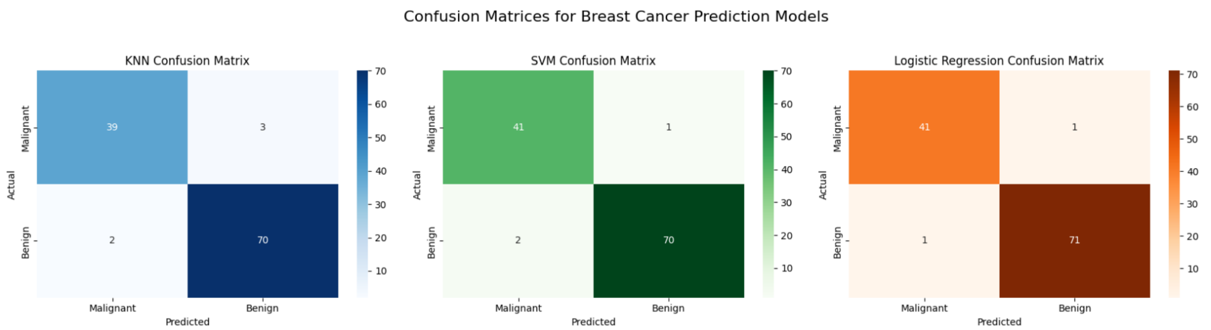
c. Results of generation on test data

```
KNN Results:

Accuracy: 0.956140350877193

Confusion Matrix:
[[39  3]
 [ 2 70]]

Classification Report:
               precision    recall  f1-score   support

Malignant (0)       0.95      0.93      0.94        42
   Benign (1)       0.96      0.97      0.97        72

     accuracy                           0.96       114
    macro avg       0.96      0.95      0.95       114
 weighted avg       0.96      0.96      0.96       114
```

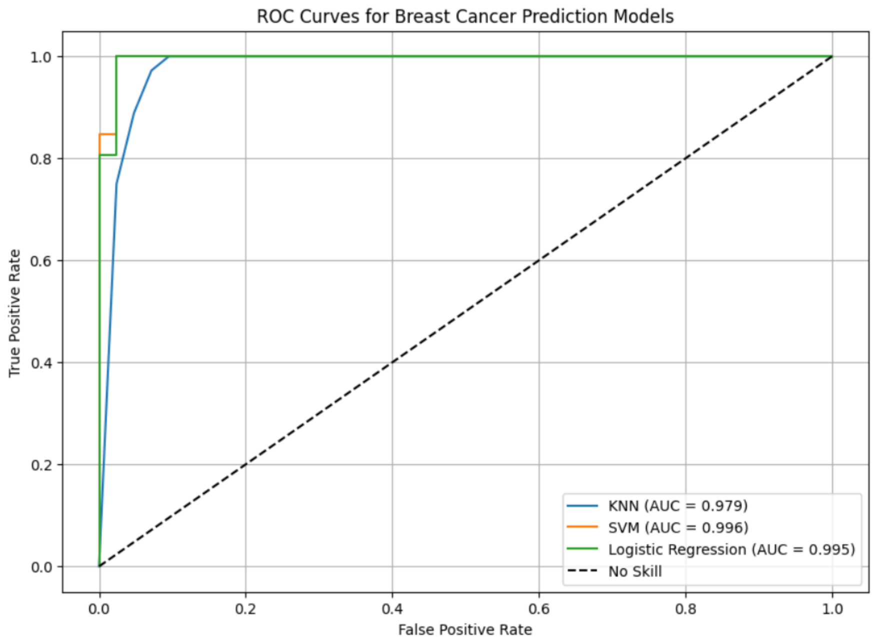
```
SVM Results:

Accuracy: 0.9736842105263158

Confusion Matrix:
[[41  1]
 [ 2 70]]

Classification Report:
               precision    recall  f1-score   support

Malignant (0)       0.95      0.98      0.96        42
   Benign (1)       0.99      0.97      0.98        72

     accuracy                           0.97       114
    macro avg       0.97      0.97      0.97       114
 weighted avg       0.97      0.97      0.97       114
```

```
Logistic Regression Results:

Accuracy: 0.9824561403508771

Confusion Matrix:
[[41  1]
 [ 1 71]]

Classification Report:
               precision    recall  f1-score   support

Malignant (0)       0.98      0.98      0.98        42
   Benign (1)       0.99      0.99      0.99        72

     accuracy                           0.98       114
    macro avg       0.98      0.98      0.98       114
 weighted avg       0.98      0.98      0.98       114
```

d. Confusion Matrices



e. ROC Curves



f. Comparison Table of Metrics

Model Performance Comparison

Model	Accuracy	Precision	Recall	F1 Score	ROC AUC
KNN	0.96	0.96	0.97	0.97	0.98
SVM	0.97	0.99	0.97	0.98	1.00
Logistic Regression	0.98	0.99	0.99	0.99	1.00

The breast cancer classification demonstrated superior performance:

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
K-Nearest Neighbours	0.96	0.96	0.97	0.97	0.98
Support Vector Machine	0.97	0.99	0.97	0.98	1.00
Logistic Regression	0.98	0.99	0.99	0.99	1.00

5.3 Comparative Analysis

- On the Diabetes dataset, the models achieved moderate accuracy (around 70–72%). Logistic Regression and SVM performed better than KNN, but none of the models achieved very high accuracy. This suggests that predicting diabetes is more challenging due to overlapping medical features and variability in patient data.
- On the Breast Cancer dataset, all models performed extremely well, with accuracy levels above 96% and Logistic Regression reaching nearly 98%. The ROC-AUC values (close to 1.0) indicated that the dataset is highly separable, making it easier for the models to distinguish between benign and malignant cases.

- Comparing both datasets, it was observed that dataset quality and feature separability strongly influence model performance. While the breast cancer dataset gave clear boundaries between classes, the diabetes dataset had more noise and overlap, leading to lower accuracy.

6. Conclusion

This study demonstrates the application of three fundamental classification algorithms to medical diagnosis tasks. The results highlight the critical importance of dataset characteristics in determining model performance, with the breast cancer dataset achieving clinically relevant accuracy levels (>96%) while diabetes prediction remained challenging with moderate performance (72%).

The workflow presented provides a foundation for medical machine learning applications, emphasizing the importance of comprehensive evaluation metrics and proper preprocessing. Future work should focus on advanced techniques including feature engineering, ensemble methods, and deep learning approaches to improve predictive performance, particularly for challenging datasets like diabetes prediction.

Future Recommendations

1. **Model Enhancement**: Implement ensemble methods and neural networks for improved performance
2. **Feature Engineering**: Apply domain knowledge to create more informative features
3. **Validation Strategy**: Employ k-fold cross-validation for robust performance estimation
4. **Clinical Integration**: Collaborate with healthcare professionals for practical deployment considerations

7. Appendix

GitHub Repo Link

https://github.com/rounakpanda/diabetes-breastcancer-classification-ml

Datasets

1. Pima Indians Diabetes Database – UCI Machine Learning Repository. Available at: https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database
2. Breast Cancer Wisconsin (Diagnostic) Dataset – UCI Machine Learning Repository. Available at: https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data

References

1. https://pmc.ncbi.nlm.nih.gov/articles/PMC10107388/
2. https://www.sciencedirect.com/science/article/pii/S1877050916302575
3. https://scikit-learn.org/stable/user_guide.html
4. https://www.kaggle.com/learn/
5. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" (O'Reilly) – Aurélien Géron

Libraries and Tools

- Python 3.12
- Google Colaboratory
- Pandas
- NumPy
- Matplotlib
- Seaborn
- Scikit-learn