DISEASE PREDICTION USING MEDICAL RECORDS.

Algorithm used: SVM (Support Vector Machine)

A

ADM Course Project Report in

partial fulfilment of the degree

Bachelor of Technology

in

Computer Science & Engineering

By

Name: K. Lasya HT No :(2303A51724)

Name: T. Pranathi HT No :(2303A51733)

Name: B. Gayathri HT NO :(2303A51717)

Under the guidance of

Dr. Jamalaiah

Assistant Professor

Submitted to

School of Computer Science and Artificial Intelligence

DEPARTMENT OFCOMPUTERSCIENCE& ENGINEERING

CERTIFICATE

This is to certify that the **Applications of Data Mining– Course Project** Report entitled **"Disease Prediction Using Medical Records."** is a record of bonafide work carried out by the student(s) "**Katta Lasya, Thotapally Pranathi, Budati Gayathri"** bearing **Hallticket No(s) 2303A51724, 2303A51733, 2303A51717,** during the academic year 2024-25 in partial fulfillment of the award of the degree of ***Bachelor of Technology*** in **Computer Science & Engineering** by the SR University, Warangal.

Supervisor	**Head of the Department**
(Dr. Jamalaiah)	(Dr. M. Sheshikala)
Assistant Professor	Professor

ORGANIZATION OF REPORT

1. Title page
2. Certificate
3. Table of Contents
4. Abstract

OBJECTIVE OF THE PROJECT

1. DEFINITIONS OF THE ELEMENTS USED IN THE PROJECT
2. DESIGN
    1. SCREENS
3. IMPLEMENTATION
    1. CODE
4. RESULTSCREENS
5. CONCLUSION

ABSTRACT

This project aims to predict the likelihood of diseases such as diabetes, heart disease, and cancer based on medical records using various machine learning techniques. The project leverages real-world datasets from UCI Machine Learning Repository and evaluates models including Decision Trees, Logistic Regression, SVM, and Neural Networks. Accuracy, precision, recall, and F1-score metrics are used for evaluation.

OBJECTIVE OF THE PROJECT

To implement a predictive system using machine learning algorithms to classify whether a patient is likely to have a particular disease (e.g., diabetes, heart disease, or cancer) based on their health metrics and records.

DEFINITIONS OF THE ELEMENTS USED

- **Dataset**: Medical datasets including attributes like glucose, blood pressure, age, insulin, etc.
- **Outcome**: Target variable indicating disease presence (0 or 1).
- **Algorithms Used**:
    - Decision Tree Classifier
    - Logistic Regression

- Support Vector Machine (SVM)
- Neural Network (MLPClassifier)

- **Metrics**: Accuracy, Precision, Recall, F1-Score

DESIGN:

2.1 SCREENS

As this is a backend machine learning model project, it does not involve front-end GUI screens. Instead, the outputs are presented through Jupyter/Colab notebooks with printed metrics and visual plots.

Simple Implementation:

```
# Load dataset import
pandas as pd
from sklearn.model_selection import train_test_split from
sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, accuracy_score
from sklearn.tree import DecisionTreeClassifier from
sklearn.linear_model import LogisticRegression from sklearn.svm
import SVC
from sklearn.neural_network import MLPClassifier
import seaborn as sns import matplotlib.pyplot as
plt

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-
indiansdiabetes.data.csv"
```

```
cols = ["Pregnancies", "Glucose", "BloodPressure", "SkinThickness", "Insulin",
        "BMI", "DiabetesPedigreeFunction", "Age", "Outcome"] data
= pd.read_csv(url, names=cols)

X = data.drop("Outcome", axis=1) y
= data["Outcome"]
X_scaled = StandardScaler().fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2)

# Decision Tree
dt = DecisionTreeClassifier().fit(X_train, y_train)
print("Decision Tree Accuracy:", accuracy_score(y_test, dt.predict(X_test)))

# Neural Network
nn = MLPClassifier(hidden_layer_sizes=(10,10), max_iter=1000).fit(X_train, y_train)
print("Neural Network Accuracy:", accuracy_score(y_test, nn.predict(X_test)))

# Logistic Regression
lr = LogisticRegression().fit(X_train, y_train)
print("Logistic Regression Accuracy:", accuracy_score(y_test, lr.predict(X_test)))

# SVM
svm = SVC().fit(X_train, y_train)
print("SVM Accuracy:", accuracy_score(y_test, svm.predict(X_test))
```

CODE:

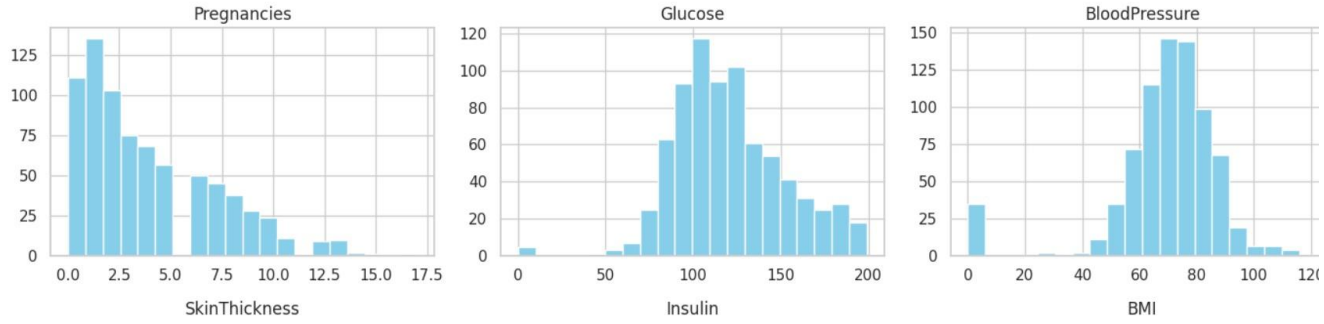
```
import pandas as pd
numpy as np import
matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split from
sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, accuracy_score
from sklearn.tree import DecisionTreeClassifier from sklearn.svm
import SVC
from sklearn.linear_model import LogisticRegression from
sklearn.neural_network import MLPClassifier

# Load dataset
url ="https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indiansdiabetes.data.csv"
columns = ["Pregnancies", "Glucose", "BloodPressure", "SkinThickness", "Insulin",
"BMI", "DiabetesPedigreeFunction", "Age", "Outcome"]

data = pd.read_csv(url, names=columns)
data.head()
```

Output:

e cell output actions

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
	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 Graphical Representation For Just Formate,



Load dataset

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indiansdiabetes.data.csv"

columns = ["Pregnancies", "Glucose", "BloodPressure", "SkinThickness",

"Insulin",

…# 5. Pairplot (can be slow with large datasets)

Uncomment the line below if you want to see relationships between features

sns.pairplot(data, hue="Outcome", corner=True) plt.show()

Load dataset of heartdisease

url ="https://archive.ics.uci.edu/static/public/45/data.csv"

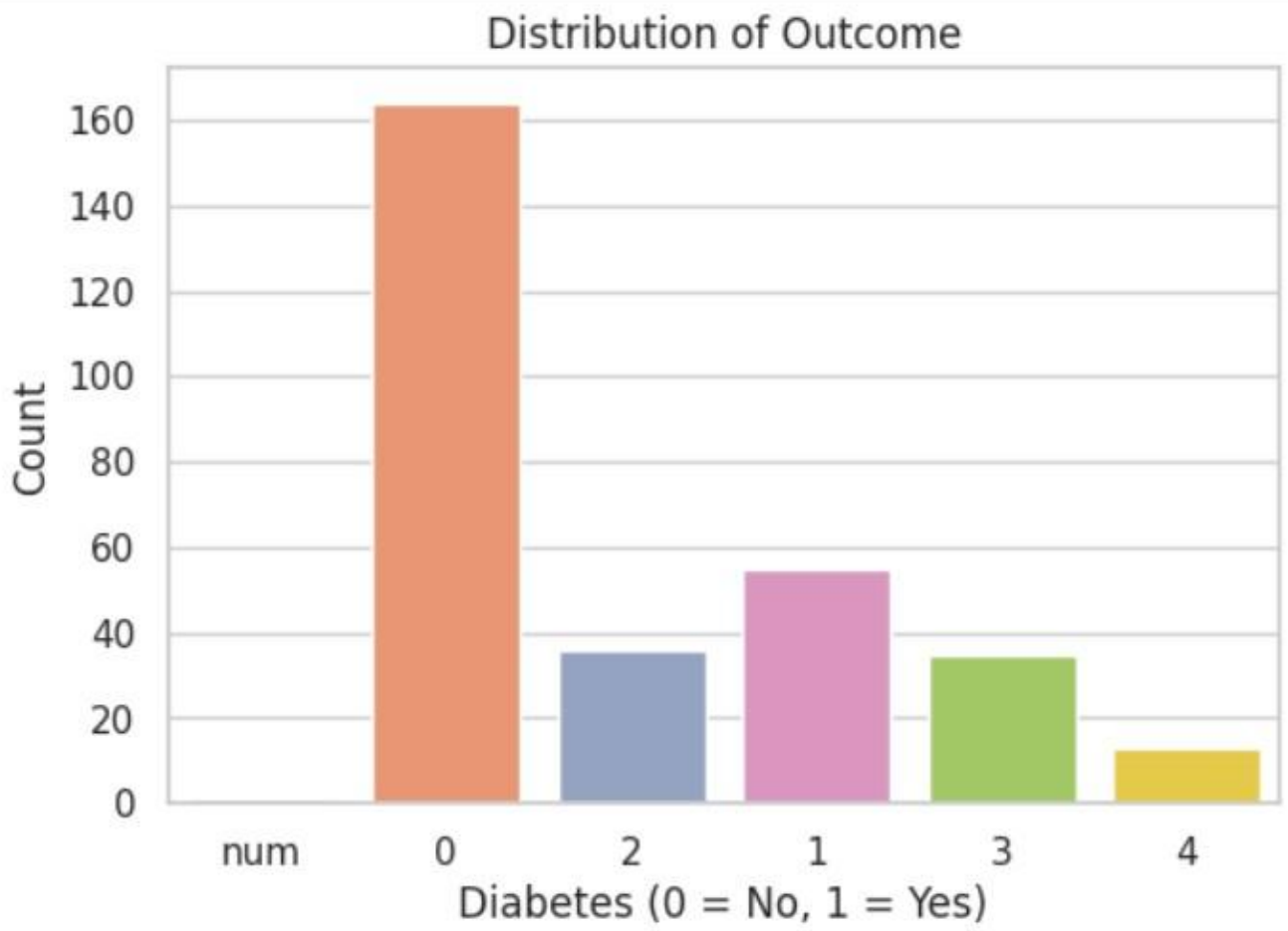
columns = ["Pregnancies", "Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI", "DiabetesPedigreeFunction", "Age", "Outcome"]

data = pd.read_csv(url, names=columns) data.head()

Output:

					Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	num
63	1	1	145	233	1	2	150	0	2.3	3	0	6	0
67	1	4	160	286	0	2	108	1	1.5	2	3	3	2
			120	229	0	2	129	1	2.6	2	2	7	1
37	1	3	130	250	0	0	187	0	3.5	3	0	3	0

The Graphical Representation For Just Formate,




```
# Load dataset of Breast Cancer
url ="https://archive.ics.uci.edu/static/public/17/data.csv"
columns = ["Pregnancies", "Glucose", "BloodPressure", "SkinThickness", "Insulin",
        "BMI", "DiabetesPedigreeFunction", "Age", "Outcome"]
data = pd.read_csv(url, names=columns) data.head()
```

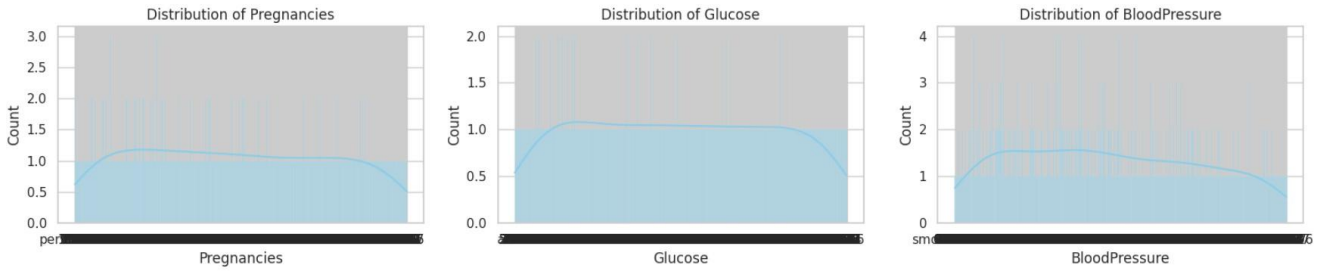
Output:

ID	radius1	texture1	perimeter1	area1	smoothness1	compactness1	concavity1	concave_points1	symmetry1	fractal_dimension1	radius2	texture2	perimeter2	area2	smoothness2
842302	17.99	10.38	122.8	1001	0.1184	0.2776	0.3001	0.1471	0.2419	0.07871	1.095	0.9053	8.589	153.4	0.006399
842517	20.57	17.77	132.9	1326	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	0.5435	0.7339	3.398	74.08	0.005225
84300903	19.69	21.25	130	1203	0.1096	0.1599	0.1974	0.1279	0.2069	0.05999	0.7456	0.7869	4.585	94.03	0.00615
84348301	11.42	20.38	77.58	386.1	0.1425	0.2839	0.2414	0.1052	0.2597	0.09744	0.4956	1.156	3.445	27.23	0.00911

compactness2	concavity2	concave_points2	symmetry2	fractal_dimension2	radius3	texture3
0.04904	0.05373	0.01587	0.03003	0.006193	25.38	17.33
0.01308	0.0186	0.0134	0.01389	0.003532	24.99	23.41
0.04006	0.03832	0.02058	0.0225	0.004571	23.57	25.53
0.07458	0.05661	0.01867	0.05963	0.009208	14.91	26.5

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
perimeter3	area3	smoothness3	compactness3	concavity3	concave_points3	symmetry3	fractal_dimension3	Diagnosis
184.6	2019	0.1622	0.6656	0.7119	0.2654	0.4601	0.1189	M
158.8	1956	0.1238	0.1866	0.2416	0.186	0.275	0.08902	M
152.5	1709	0.1444	0.4245	0.4504	0.243	0.3613	0.08758	M
98.87	567.7	0.2098	0.8663	0.6869	0.2575	0.6638	0.173	M

The Graphical Representation For Just Formate,



Load dataset of Kidney. url

="https://archive.ics.uci.edu/static/public/336/data.csv"

columns = ["Pregnancies", "Glucose", "BloodPressure", "SkinThickness", "Insulin",

"BMI", "DiabetesPedigreeFunction", "Age", "Outcome"]

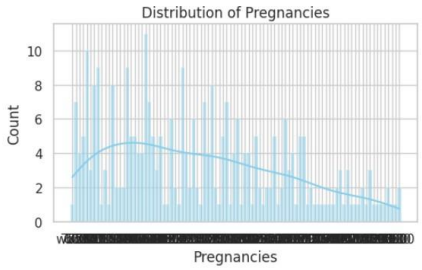
data = pd.read_csv(url, names=columns) data.head()

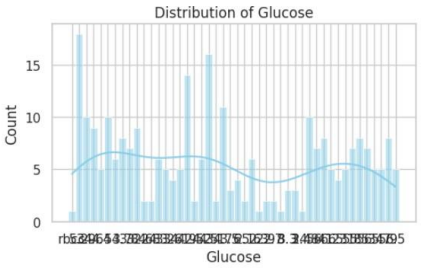
Output:

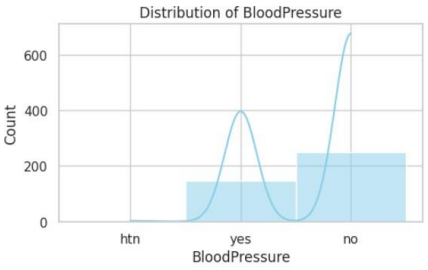
															Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin
age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	bu	sc	sod	pot	hemo	pcv	wbcc	rbcc	htn	dm	cad
48	80	1.02	1	0	NaN	normal	notpresent	notpresent	121	36	1.2	NaN	NaN	15.4	44	7800	5.2	yes	yes	no
7	50	1.02	4	0	NaN	normal	notpresent	notpresent	NaN	18	0.8	NaN	NaN	11.3	38	6000	NaN	no	no	no
62	80	1.01	2	3	normal	normal	notpresent	notpresent	423	53	1.8	NaN	NaN	9.6	31	7500	NaN	no	yes	no
48	70	1.005	4	0	normal	abnormal	present	notpresent	117	56	3.8	111	2.5	11.2	32	6700	3.9	yes	no	no

BMI	DiabetesPedigreeFunction	Age	Outcome
appet	pe	ane	class
good	no	no	ckd
good	no	no	ckd
poor	no	yes	ckd
poor	yes	yes	ckd

The Graphical Representation For Just Formate,







X = data.drop("Outcome", axis=1)

y = data["Outcome"] 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) dt = DecisionTreeClassifier()

```
dt.fit(X_train, y_train) y_pred_dt
= dt.predict(X_test)
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_dt))
print(classification_report(y_test, y_pred_dt))
```

Output:

```
Decision Tree Accuracy: 0.7337662337662337
              precision    recall  f1-score   support

           0       0.82      0.76      0.79        99
           1       0.61      0.69      0.65        55

    accuracy                           0.73       154
   macro avg       0.71      0.72      0.72       154
weighted avg       0.74      0.73      0.74       154
```

```
nn = MLPClassifier(hidden_layer_sizes=(10,10), max_iter=1000)
nn.fit(X_train, y_train) y_pred_nn = nn.predict(X_test)
print("Neural Network Accuracy:", accuracy_score(y_test, y_pred_nn))
print(classification_report(y_test, y_pred_nn))
```

Output:

```
Neural Network Accuracy: 0.7467532467532467
              precision    recall  f1-score   support

           0       0.81      0.80      0.80        99
           1       0.64      0.65      0.65        55

    accuracy                           0.75       154
   macro avg       0.72      0.73      0.73       154
weighted avg       0.75      0.75      0.75       154
```

```
lr = LogisticRegression()
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)
print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred_lr))
print(classification_report(y_test, y_pred_lr))
```

Output:

```
Logistic Regression Accuracy: 0.7532467532467533
              precision    recall  f1-score   support

           0       0.81      0.80      0.81        99
           1       0.65      0.67      0.66        55

    accuracy                           0.75       154
   macro avg       0.73      0.74      0.73       154
weighted avg       0.76      0.75      0.75       154
```

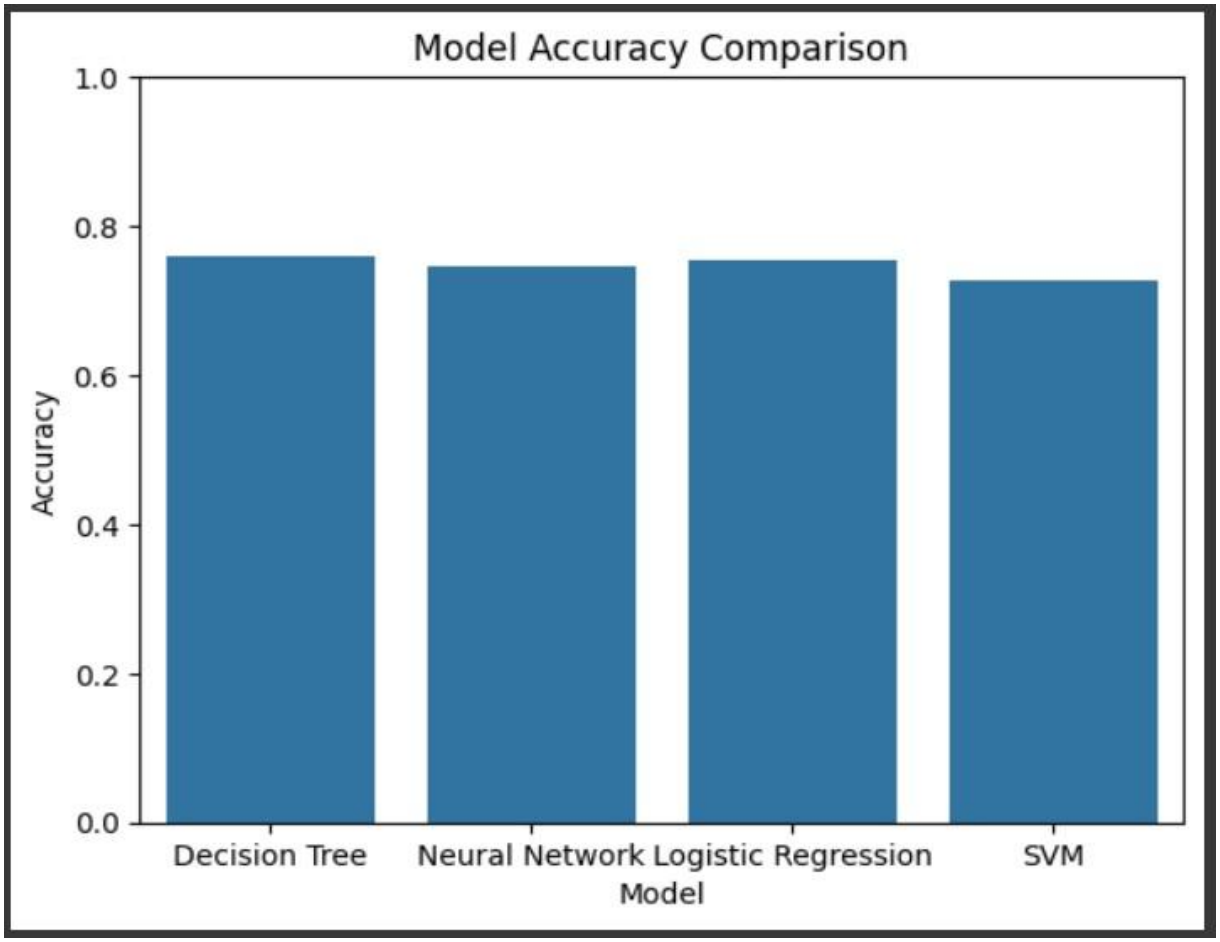
```
svm = SVC() svm.fit(X_train,
y_train) y_pred_svm =
svm.predict(X_test)
print("SVM Accuracy:", accuracy_score(y_test, y_pred_svm))
print(classification_report(y_test, y_pred_svm))
```

Output:

```
SVM Accuracy: 0.7272727272727273
              precision    recall  f1-score   support

           0       0.77      0.82      0.79        99
           1       0.63      0.56      0.60        55

    accuracy                           0.73       154
   macro avg       0.70      0.69      0.70       154
weighted avg       0.72      0.73      0.72       154
```

```
results = {
    "Model": ["Decision Tree", "Neural Network", "Logistic Regression", "SVM"],
    "Accuracy": [
        accuracy_score(y_test, y_pred_dt),
accuracy_score(y_test, y_pred_nn),
accuracy_score(y_test, y_pred_lr),
accuracy_score(y_test, y_pred_svm)
    ]
}
df_results = pd.DataFrame(results)
sns.barplot(x="Model", y="Accuracy", data=df_results)
plt.title("Model Accuracy Comparison") plt.ylim(0, 1)
plt.show()
```

Output:



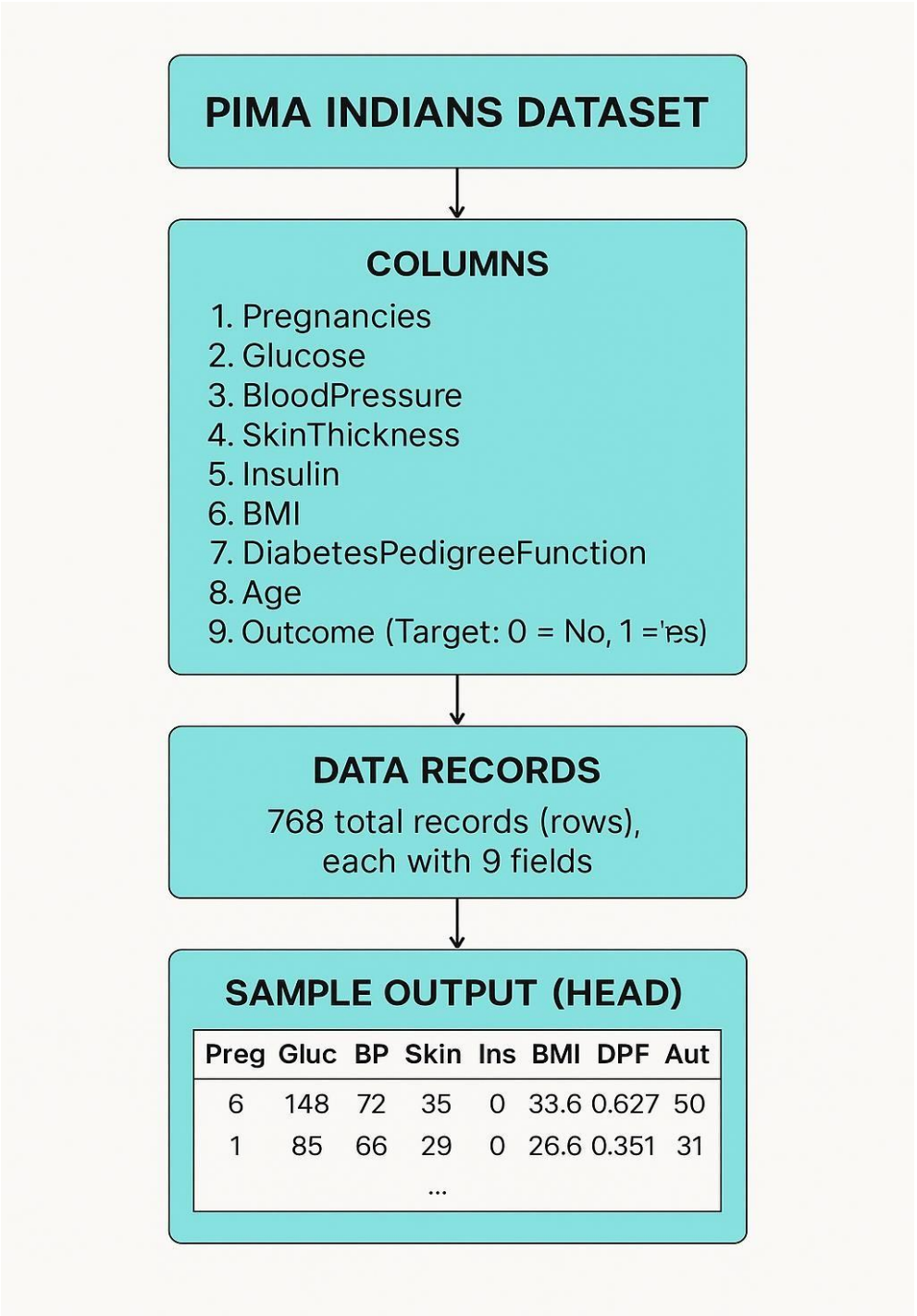
RESULT SCREENS:

Model	Accuracy
Decision Tree	73.37%
Neural Network	74.68%
Logistic Regression	75.32%
SVM	72.73%

CONCLUSION:

From the analysis and model ing of the dataset using different machine learning techniques, Logistic Regression provided the highest accuracy of 75.32%. The project shows that patient health metrics can be effectively used to build predictive models for early diagnosis of diseases like diabetes. Future work can include using ensemble methods or deep learning for improved results.

BLOCK DIAGRAM:


PIMA INDIANS DATASET
COLUMNS
1. Pregnancies
2. Glucose
3. BloodPressure
4. SkinThickness
5. Insulin
6. BMI
7. DiabetesPedigreeFunction
8. Age
9. Outcome (Target: 0 = No, 1 = Yes)
DATA RECORDS
768 total records (rows),
each with 9 fields
SAMPLE OUTPUT (HEAD)
Preg Gluc BP Skin Ins BMI DPF Aut
6 148 72 35 0 33.6 0.627 50
1 85 66 29 0 26.6 0.351 31
...

Thank You