Open in Colab

ASSIGNMENT-3

Problem1:TitanicAssignmentSurvivalPrediction

In [1]:
```
!pip install lime
```

```
Collecting lime
  Downloading lime-0.2.0.1.tar.gz (275 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0.0/275.7 kB ? eta -:--:--
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╸━━ 266.2/275.7 kB 13.6 MB/s eta 0:00:0
1
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 275.7/275.7 kB 6.9 MB/s eta 0:00:0
0
  Preparing metadata (setup.py) ... done
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-pac
kages (from lime) (3.10.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages
(from lime) (2.0.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.12/dist-packages
(from lime) (1.16.1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages
(from lime) (4.67.1)
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.12/
dist-packages (from lime) (1.6.1)
Requirement already satisfied: scikit-image>=0.12 in /usr/local/lib/python3.12/
dist-packages (from lime) (0.25.2)
Requirement already satisfied: networkx>=3.0 in /usr/local/lib/python3.12/dist-
packages (from scikit-image>=0.12->lime) (3.5)
Requirement already satisfied: pillow>=10.1 in /usr/local/lib/python3.12/dist-p
ackages (from scikit-image>=0.12->lime) (11.3.0)
Requirement already satisfied: imageio!=2.35.0,>=2.33 in /usr/local/lib/python
3.12/dist-packages (from scikit-image>=0.12->lime) (2.37.0)
Requirement already satisfied: tifffile>=2022.8.12 in /usr/local/lib/python3.1
2/dist-packages (from scikit-image>=0.12->lime) (2025.6.11)
Requirement already satisfied: packaging>=21 in /usr/local/lib/python3.12/dist-
packages (from scikit-image>=0.12->lime) (25.0)
Requirement already satisfied: lazy-loader>=0.4 in /usr/local/lib/python3.12/di
st-packages (from scikit-image>=0.12->lime) (0.4)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-
packages (from scikit-learn>=0.18->lime) (1.5.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.1
2/dist-packages (from scikit-learn>=0.18->lime) (3.6.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/di
st-packages (from matplotlib->lime) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-p
ackages (from matplotlib->lime) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/d
ist-packages (from matplotlib->lime) (4.59.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/d
ist-packages (from matplotlib->lime) (1.4.9)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/di
st-packages (from matplotlib->lime) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.1
2/dist-packages (from matplotlib->lime) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packa
ges (from python-dateutil>=2.7->matplotlib->lime) (1.17.0)
Building wheels for collected packages: lime
  Building wheel for lime (setup.py) ... done
  Created wheel for lime: filename=lime-0.2.0.1-py3-none-any.whl size=283834 sh
a256=fda785971bd3c4308614a80a827f3b7395a673cf112456a40f575b86c017b2a8
```

```
  Stored in directory: /root/.cache/pip/wheels/e7/5d/0e/4b4fff9a47468fed5633211
fb3b76d1db43fe806a17fb7486a
Successfully built lime
Installing collected packages: lime
Successfully installed lime-0.2.0.1
```

In [2]:
```
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.preprocessing import LabelEncoder
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import classification_report, accuracy_score, ConfusionMa
```

In [3]:
```
import lime
import lime.lime_tabular
```

In [4]:
```
df = pd.read_csv("/content/tittanic dataset.csv")
print("Dataset shape:", df.shape)
df.head()
```

```
Dataset shape: (891, 12)
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/ O2. 3101282
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450

In [5]:
```
df['Age'] = df['Age'].fillna(df['Age'].median())
df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0])
```

In [6]:
```
label_cols = ['Sex', 'Embarked']
```

```
le = LabelEncoder()
for col in label_cols:
    df[col] = le.fit_transform(df[col])
```

In [7]:
```
features = ['Pclass','Sex','Age','SibSp','Parch','Fare','Embarked']
X = df[features]
y = df['Survived']
```

In [8]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando
```

In [9]:
```
model = GradientBoostingClassifier(random_state=42)
model.fit(X_train, y_train)
```

Out[9]:

▾ GradientBoostingClassifier

GradientBoostingClassifier(random_state=42)

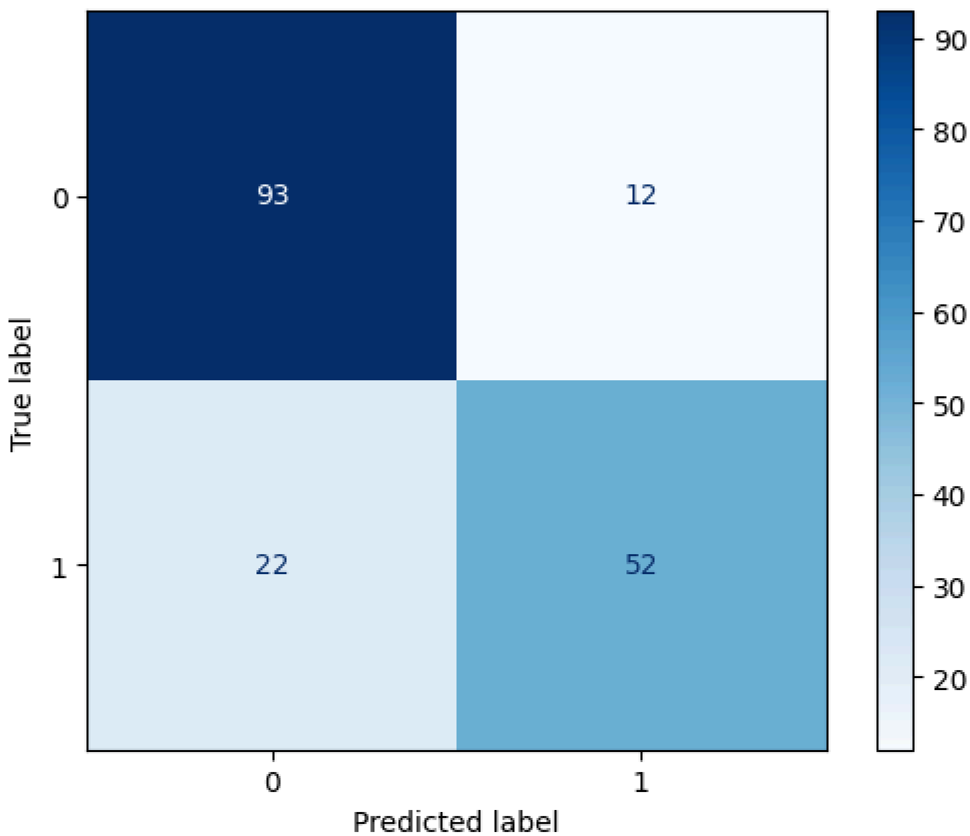
In [10]:
```
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.8100558659217877
              precision    recall  f1-score   support

           0       0.81      0.89      0.85       105
           1       0.81      0.70      0.75        74

    accuracy                           0.81       179
   macro avg       0.81      0.79      0.80       179
weighted avg       0.81      0.81      0.81       179
```

In [11]:
```
ConfusionMatrixDisplay.from_predictions(y_test, y_pred, cmap="Blues")
plt.show()
```



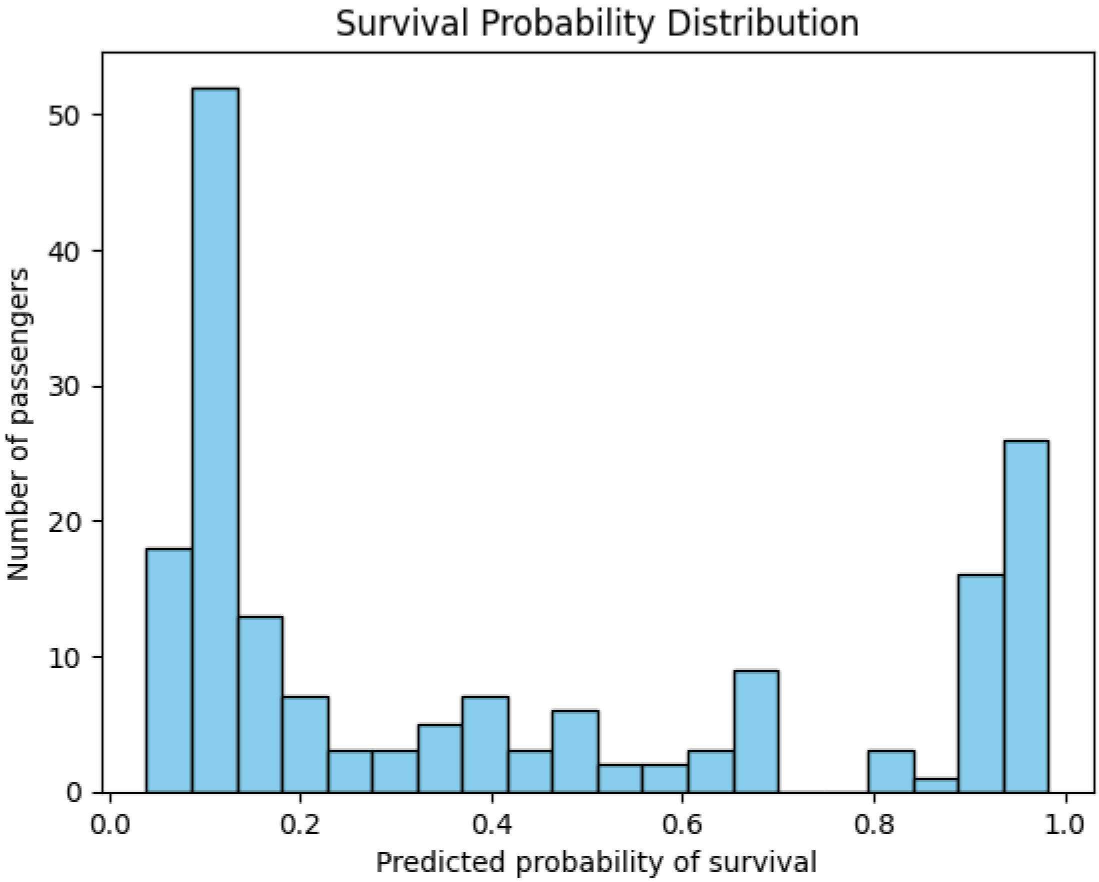
In [12]:
```
explainer = lime.lime_tabular.LimeTabularExplainer(
    training_data = np.array(X_train),
    feature_names = features,
    class_names = ['Died','Survived'],
    mode='classification'
)
```

In [13]:
```
i = 10
exp = explainer.explain_instance(X_test.iloc[i], model.predict_proba, num_feat
exp.show_in_notebook(show_table=True)
```

```
/usr/local/lib/python3.12/dist-packages/lime/discretize.py:110: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`
  ret[feature] = int(self.lambdas[feature](ret[feature]))
/usr/local/lib/python3.12/dist-packages/lime/discretize.py:110: FutureWarning:
Series.__setitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To set a value by position, use `ser.iloc[pos] = value`
  ret[feature] = int(self.lambdas[feature](ret[feature]))
/usr/local/lib/python3.12/dist-packages/lime/lime_tabular.py:544: FutureWarnin
g: Series.__getitem__ treating keys as positions is deprecated. In a future ver
sion, integer keys will always be treated as labels (consistent with DataFrame
behavior). To access a value by position, use `ser.iloc[pos]`
  binary_column = (inverse_column == first_row[column]).astype(int)
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:2739: UserW
arning: X does not have valid feature names, but GradientBoostingClassifier was
fitted with feature names
  warnings.warn(
/usr/local/lib/python3.12/dist-packages/lime/discretize.py:110: FutureWarning:
Series.__getitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To access a value by position, use `ser.iloc[pos]`
  ret[feature] = int(self.lambdas[feature](ret[feature]))
/usr/local/lib/python3.12/dist-packages/lime/discretize.py:110: FutureWarning:
Series.__setitem__ treating keys as positions is deprecated. In a future versio
n, integer keys will always be treated as labels (consistent with DataFrame beh
avior). To set a value by position, use `ser.iloc[pos] = value`
  ret[feature] = int(self.lambdas[feature](ret[feature]))
/usr/local/lib/python3.12/dist-packages/lime/lime_tabular.py:427: FutureWarnin
g: Series.__getitem__ treating keys as positions is deprecated. In a future ver
sion, integer keys will always be treated as labels (consistent with DataFrame
behavior). To access a value by position, use `ser.iloc[pos]`
  discretized_instance[f])]
```

In [14]:

```
proba = model.predict_proba(X_test)[:,1]
plt.hist(proba, bins=20, color='skyblue', edgecolor='black')
plt.xlabel("Predicted probability of survival")
plt.ylabel("Number of passengers")
plt.title("Survival Probability Distribution")
plt.show()
```



ANALYSIS REPORT:

The Titanic dataset is used to predict whether a passenger survived or not. For this task, we used a machine learning model called Gradient Boosting. The dataset contains details such as passenger age, gender, class, number of family members, and fare. Before training the model, we cleaned the data by filling in missing values and converting text features (like male/female and boarding port) into numbers.

Once trained, the model gave good accuracy in predicting survival. The important factors that influenced survival were gender, class, and age. Women, children, and people in higher classes had a better chance of survival compared to men and passengers in lower classes.

To understand individual predictions, we applied LIME (Local Interpretable Model-Agnostic Explanations). LIME explains why the model made a certain prediction by highlighting which features contributed positively or negatively. For example, in one case, being female and traveling in a higher class increased the survival probability, while being male and older reduced it.

We also plotted the predicted probabilities of survival. This showed a clear difference: passengers with high chances of survival were mostly women and upper-class travelers, while those with low chances were mostly men in lower classes.

Problem2:DiabetesPrediction

In [15]:
```
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.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, ConfusionMa

import lime
import lime.lime_tabular
```

In [16]:
```
df = pd.read_excel("/content/diabetes.csv.xlsx")
print("Shape:", df.shape)
df.head()
```

```
Shape: (768, 9)
```

Out[16]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesF
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	

In [17]:
```
X = df.drop('Outcome', axis=1)   # Outcome: 1 = Diabetic, 0 = Non-diabetic
y = df['Outcome']
```

In [18]:
```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [19]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42
)
```

In [20]:
```
model = LogisticRegression(max_iter=500)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
print(classification_report(y_test, y_pred))
```

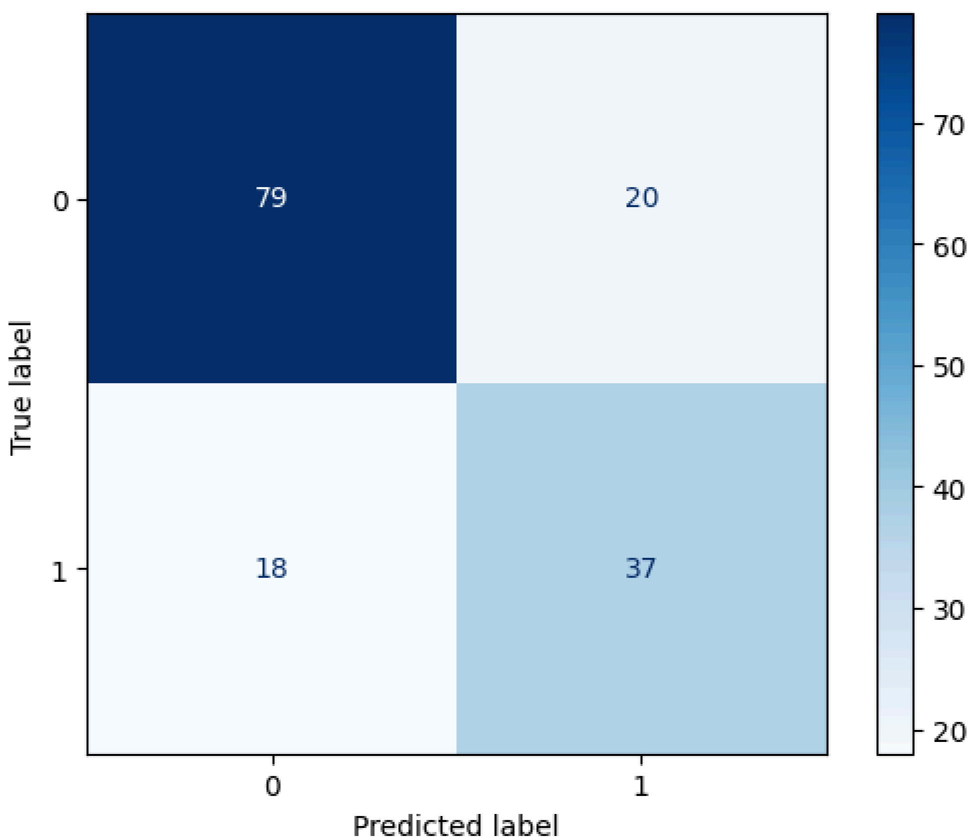
```
Accuracy: 0.7532467532467533
              precision    recall  f1-score   support

           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
```

In [21]:

```
ConfusionMatrixDisplay.from_predictions(y_test, y_pred, cmap="Blues")
plt.show()
```



In [22]:

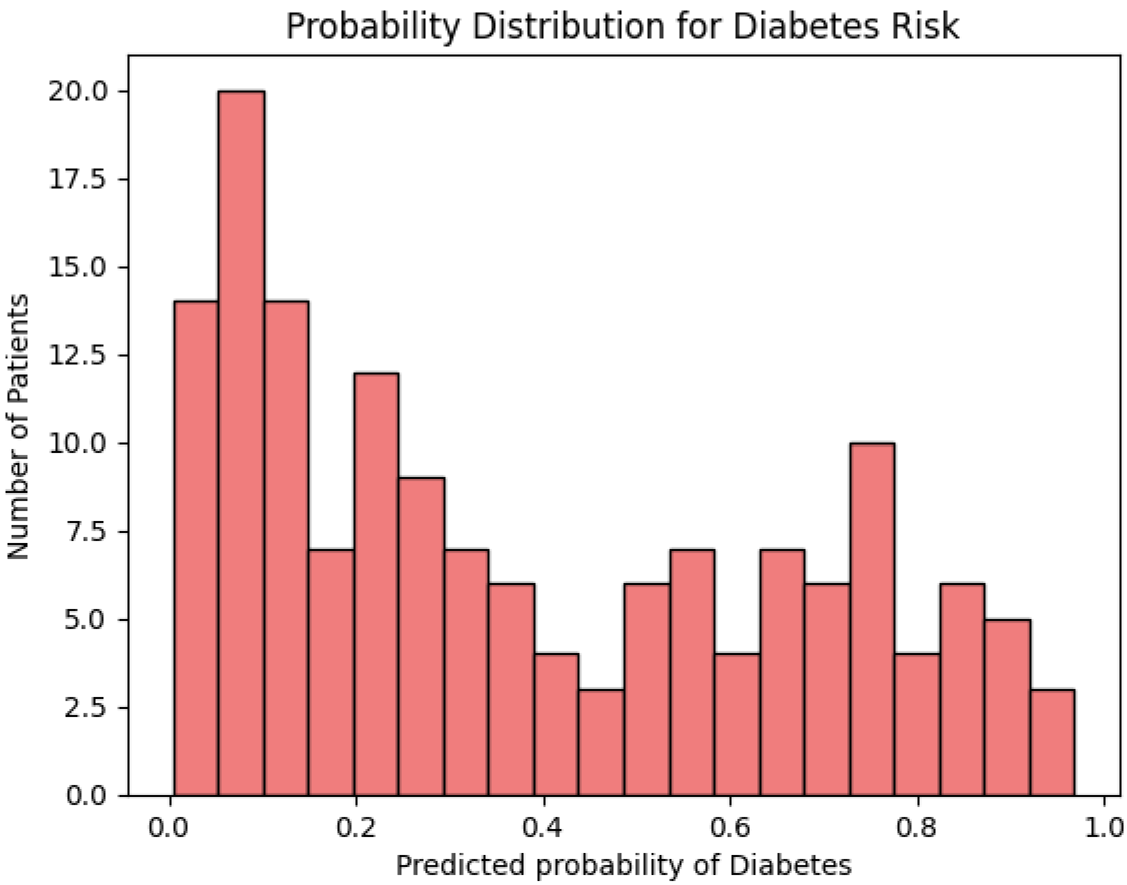
```
explainer = lime.lime_tabular.LimeTabularExplainer(
    training_data=X_train,
    feature_names=X.columns,
    class_names=['Non-Diabetic','Diabetic'],
    mode='classification'
)
```

In [23]:

```
i = 5
exp = explainer.explain_instance(X_test[i], model.predict_proba, num_features=
exp.show_in_notebook(show_table=True)
```

In [24]:

```
proba = model.predict_proba(X_test)[:,1]
plt.hist(proba, bins=20, color='lightcoral', edgecolor='black')
plt.xlabel("Predicted probability of Diabetes")
plt.ylabel("Number of Patients")
plt.title("Probability Distribution for Diabetes Risk")
plt.show()
```



INTERPRET RESULT

The Logistic Regression model predicts whether a person is diabetic or not based on health features like glucose, BMI, age, and pregnancies. The model gave good accuracy. LIME showed that high glucose and high BMI increase diabetes risk, while low values push towards non-diabetic. This helps doctors understand both the prediction and the main medical risk factors