```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt from sklearn.model_selection
import train_test_split from sklearn.linear_model
import LogisticRegression from sklearn.metrics
import classification_report, confusion_matrix
df = pd.read_csv('/content/Social_Network_Ads.csv')
X = X.reshape(-1, 1)
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt from sklearn.model_selection
import train_test_split from sklearn.linear_model
import LogisticRegression from sklearn.metrics
import classification_report, confusion_matrix
df = pd.read_csv('/content/Social_Network_Ads.csv')
X = df.iloc[:, 1].values
y = df.iloc[:, -1].values
X = X.reshape(-1, 1)
df.head()
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt from sklearn.model_selection
import train_test_split from sklearn.linear_model
import LogisticRegression from sklearn.metrics
import classification_report, confusion_matrix
df = pd.read_csv('/content/Social_Network_Ads.csv')
 X = df.iloc[:, 1].values
y = df.iloc[:, -1].values
X = X.reshape(-1, 1)
df.head()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
print(X_train)
print(X_test)
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
```

```
le = LabelEncoder()
X_train[:, 0] = le.fit_transform(X_train[:, 0])
X_test[:, 0] = le.transform(X_test[:, 0])
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print(X_train)
print(X_test)
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
print(classifier.predict(sc.transform([[87000]])))
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1)
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
accuracy_score(y_test, y_pred)
print(classification_report(y_test, y_pred))
!pip install -U scikit-learn
from sklearn.metrics import PrecisionRecallDisplay, precision_recall_curve
import matplotlib.pyplot as plt
precision, recall, _ = precision_recall_curve(classifier.predict(X_test), y_test)
disp = PrecisionRecallDisplay(precision=precision, recall=recall)
disp.plot()
plt.title('Precision-Recall curve for Logistic Regression')
plt.show()
from sklearn.metrics import roc_curve
pred_prob1 = classifier.predict_proba(X_test)
fpr1, tpr1, thresh1 = roc_curve(y_test, pred_prob1[:,1], pos_label=1)
random_probs = [0 for i in range(len(y_test))]
p_fpr, p_tpr, _ = roc_curve(y_test, random_probs, pos_label=1)
!pip install seaborn
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_style('darkgrid') # or any other seaborn style you prefer
plt.plot(fpr1, tpr1, linestyle='--',color='orange', label='Logistic Regression')
plt plot(p fpr p tpr linestyle='--' color='blue')
plt.xlabel('False Positive Rate')
plt.legend(loc='best')
```

```
plt.savefig('ROC',dpi=300)
plt.show();
DF = pd.read_csv('/content/diabetes.csv')
print(DF.head())
import pandas as pd
null_values = DF.isnull().sum()
print("Null values in each column:\n", null_values)
if DF.isnull().values.any():
print("\nThe DataFrame contains null values.")
else:
print("\nThe DataFrame does not contain null values.")
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
X = DF.drop('Outcome', axis=1) # Features
y = DF['Outcome'] # Target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
import pandas as pd
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, roc_curve, auc
import matplotlib.pyplot as plt
fpr, tpr, thresholds = roc_curve(y_test, y_pred) # Get values for plotting
roc_auc = auc(fpr, tpr) # Calculate Area Under the Curve (AUC)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--') # Diagonal line (random classifier)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
X = df.iloc[:, :-1].values # considering age,estimated salary
X = df.iloc[:, :-1].values
# Select the last column of 'df' to create 'y'
y = df.iloc[:, -1].values
df.head()
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
softmax_reg = LogisticRegression(multi_class='multinomial', # switch to Softmax Regression
solver='lbfgs', # handle multinomial loss, L2 penalty
C=10)
softmax_reg = LogisticRegression(multi_class='multinomial', # switch to Softmax Regression
solver='lbfgs', # handle multinomial loss, L2 penalty
C=10)
softmax_reg.fit(X_train, y_train) # Add this line to train the model
softmax_reg.predict(sc.transform([[30, 87000, 1]]))
softmax_reg.predict_proba(sc.transform([[30, 87000, 0]]))
from sklearn.datasets import make_classification
X, y = make_classification(n_samples=200, n_features=2, n_informative=2, n_redundant=0, n_classes=2, random_state=1)
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
```

```
from mlxtend.plotting import plot_decision_regions
from sklearn.linear_model import LogisticRegression # Import LogisticRegression
# Install mlxtend if necessary
!pip install mlxtend
s = gridspec.GridSpec(3, 2)
fig = plt.figure(figsize=(14,10))
label = 'Logistic Regression'
# Create 'clf' by instantiating LogisticRegression
clf = LogisticRegression()
# Fit(X, y) on 'clf'
clf.fit(X, y)
fig = plot_decision_regions(X=X, y=y, clf=clf, legend=2)
plt.title(label)
plt.show()
```