
Figure 1 consists of two bar charts. The left chart is titled 'General population' and the right chart is titled 'Respondents personally affected by the economic crisis'. Both charts show the percentage of respondents for different levels of agreement with the statement 'The government should do more to help people who are struggling financially'. The levels of agreement are 'Strongly agree', 'Agree', 'Disagree', and 'Strongly disagree'. The y-axis represents the percentage of respondents, ranging from 0 to 100. The x-axis represents the levels of agreement. In both charts, the 'Strongly agree' and 'Agree' categories show high percentages, while 'Disagree' and 'Strongly disagree' show much lower percentages. The 'Agree' category is the most prominent in both charts.

Level of Agreement	General population (%)	Respondents personally affected by the economic crisis (%)
Strongly agree	~15	~15
Agree	~65	~65
Disagree	~15	~15
Strongly disagree	~5	~5



Age Group	Percentage
18-24	~15%
25-34	~25%
35-44	~35%
45-54	~45%
55-64	~55%
65-74	~65%
75-84	~75%
85+	~85%

```

1  # Import the necessary libraries
2  import pandas as pd
3  import numpy as np
4  import matplotlib.pyplot as plt
5  import seaborn as sns
6  from sklearn.preprocessing import StandardScaler
7  from sklearn.model_selection import train_test_split
8  from sklearn.metrics import accuracy_score, confusion_matrix, roc_auc_score
9  from sklearn.svm import SVC
10 from sklearn.ensemble import RandomForestClassifier
11 from sklearn.linear_model import LogisticRegression
12 from sklearn.metrics import classification_report
13
14 # Load the dataset
15 data = pd.read_csv('data.csv')
16
17 # Check the shape of the dataset
18 print(data.shape)
19
20 # Display the first few rows of the dataset
21 print(data.head())
22
23 # Check for missing values
24 print(data.isnull().sum())
25
26 # Drop missing values
27 data = data.dropna()
28
29 # Check the distribution of the target variable
30 print(data['target'].value_counts())
31
32 # Split the data into training and testing sets
33 X = data.drop('target', axis=1)
34 y = data['target']
35 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
36
37 # Standardize the features
38 scaler = StandardScaler()
39 X_train = scaler.fit_transform(X_train)
40 X_test = scaler.transform(X_test)
41
42 # Train the Logistic Regression model
43 lr = LogisticRegression()
44 lr.fit(X_train, y_train)
45
46 # Predict the target variable for the test set
47 y_pred_lr = lr.predict(X_test)
48
49 # Calculate the accuracy score for the Logistic Regression model
50 accuracy_lr = accuracy_score(y_test, y_pred_lr)
51
52 # Print the accuracy score
53 print('Accuracy of Logistic Regression: {}'.format(accuracy_lr))
54
55 # Train the Random Forest Classifier model
56 rf = RandomForestClassifier()
57 rf.fit(X_train, y_train)
58
59 # Predict the target variable for the test set
60 y_pred_rf = rf.predict(X_test)
61
62 # Calculate the accuracy score for the Random Forest Classifier model
63 accuracy_rf = accuracy_score(y_test, y_pred_rf)
64
65 # Print the accuracy score
66 print('Accuracy of Random Forest Classifier: {}'.format(accuracy_rf))
67
68 # Train the Support Vector Machine (SVC) model
69 svc = SVC()
70 svc.fit(X_train, y_train)
71
72 # Predict the target variable for the test set
73 y_pred_svc = svc.predict(X_test)
74
75 # Calculate the accuracy score for the SVC model
76 accuracy_svc = accuracy_score(y_test, y_pred_svc)
77
78 # Print the accuracy score
79 print('Accuracy of Support Vector Machine (SVC): {}'.format(accuracy_svc))
80
81 # Generate a confusion matrix for the SVC model
82 cm = confusion_matrix(y_test, y_pred_svc)
83
84 # Print the confusion matrix
85 print('Confusion Matrix for SVC:')
86 print(cm)
87
88 # Generate a ROC curve for the SVC model
89 fpr, tpr, _ = roc_curve(y_test, svc.decision_function(X_test))
90
91 # Plot the ROC curve
92 plt.figure()
93 plt.plot(fpr, tpr, label='SVC')
94 plt.plot([0, 1], [0, 1], label='Random')
95 plt.xlabel('False Positive Rate')
96 plt.ylabel('True Positive Rate')
97 plt.title('ROC Curve for SVC')
98 plt.legend()
99 plt.show()
100
101 # Generate a classification report for the SVC model
102 print('Classification Report for SVC:')
103 print(classification_report(y_test, y_pred_svc))

```