Assignment #	3

By

NAME : Ishmal shahid

ROLL NO #: 1220100981

BS SE (4th)

Fall 2022

COURSE TITLE: ARTIFICAL INTELLIGENCE

SUBMITTED TO : SIR ZUBAR

Department of COMPUTER SCIENCE
International Institute of Science, Arts and Technology (IISAT), Gujranwala

TASK NO 1:

```
import pandas as pd

from matplotlib import pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Read the data

df = pd.read_csv("insurance_data.csv")

# Visualize the data

plt.figure(figsize=(8, 6))

plt.scatter(df.age, df.bought_insurance, marker='+', color='red')

plt.xlabel('Age')

plt.ylabel('Bought Insurance')

plt.title('Insurance Data: Age vs. Bought Insurance')

plt.show()

# Split the data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(df[['age']], df.bought_insurance, train_size=0.8,
random_state=42)

# Create and fit the logistic regression model

model = LogisticRegression()

model.fit(X_train, y_train)

# Make predictions on the test data

y_predicted = model.predict(X_test)



# Calculate the accuracy metrics
```

```
accuracy = accuracy_score(y_test, y_predicted)

precision = precision_score(y_test, y_predicted)

recall = recall_score(y_test, y_predicted)

f1 = f1_score(y_test, y_predicted)

print("Accuracy:", accuracy)

print("Precision:", precision)

print("Recall:", recall)

print("F1-Score:", f1)

# Visualize the logistic regression fit

plt.figure(figsize=(8, 6))

plt.scatter(df.age, df.bought_insurance, marker='+', color='red', label='Data points')

plt.plot(X_test, model.predict_proba(X_test)[:, 1], color='blue', label='Logistic regression fit')

plt.xlabel('Age')

plt.ylabel('Probability of Buying Insurance')

plt.title('Logistic Regression Fit')

plt.legend()

plt.show()

# Make predictions for a different age value

new_age = 35

predicted_value = model.predict([[new_age]])

print(f"Predicted value for age {new_age}: {predicted_value}")
```

Explanation:

- **Visualizing Data**: The scatter plot shows how insurance purchase varies with age.
- **Training Logistic Regression Model**: `LogisticRegression()` is used to fit the model on the training data (`X_train`, `y_train`).

- **Predictions**: `model.predict()` predicts whether insurance was bought (`y_predicted`) based on age (`X_test`).
- **Metrics Calculation**: `accuracy_score`, `precision_score`, `recall_score`, and `f1_score` calculate the respective metrics using `y_test` (actual values) and `y_predicted` (predicted values).
- **Visualization of Model Fit**: Shows the logistic regression curve over the scatter plot to visualize how the model predicts the probability of buying insurance based on age.
- **Making Predictions**: Demonstrates making predictions for a new age value (`new_age = 35`) using `model.predict()`.

Interpretation:

- **Accuracy**: Tells you the overall correctness of the model's predictions.
- **Precision**: Indicates how precise the model is in predicting individuals who actually bought insurance.
- **Recall**: Indicates how well the model captures individuals who actually bought insurance.
- **F1-Score**: Provides a balance between precision and recall.

  VALUE

  Accuracy: 1.0

  Precision: 1.0

  Recall: 1.0

  F1-Score: 1.0

TASK NO 2:

The confusion matrix is a table that is used to describe the performance of a classification model. It compares the actual target values with the predictions made by the model.

- **TN (True Negative)**: The number of negative instances correctly classified as negative.
- **FP (False Positive)**: The number of negative instances incorrectly classified as positive.
- **FN (False Negative)**: The number of positive instances incorrectly classified as negative.
- **TP (True Positive)**: The number of positive instances correctly classified as positive.

How the Confusion Matrix Helps

The confusion matrix helps to understand the performance of the model in more detail than accuracy alone. It provides insight into the types of errors the model is making:

- **High TP and TN**: Indicates good model performance.
- **High FP**: Indicates that the model is incorrectly predicting positive instances.
- **High FN**: Indicates that the model is incorrectly predicting negative instances.

For example, consider a medical diagnosis model:

- **FN (False Negative)**: A patient with a disease is predicted to be healthy. This is a serious error.
- **FP (False Positive)**: A healthy patient is predicted to have a disease. This leads to unnecessary further testing but is less critical than an FN.

Understanding the confusion matrix helps to balance precision and recall based on the application requirements, such as ensuring low FN for medical diagnoses or low FP for spam detection.

```
# Create and print the confusion matrix

conf_matrix = confusion_matrix(y_test, y_predicted)

print("Confusion Matrix:")

print(conf_matrix)

# Interpret the confusion matrix

tn, fp, fn, tp = conf_matrix.ravel()

print(f"True Negatives (TN): {tn}")

print(f"False Positives (FP): {fp}")

print(f"False Negatives (FN): {fn}")

print(f"True Positives (TP): {tp}")

# Explanation

print("\nExplanation:")

print(f"Accuracy: {(tn + tp) / (tn + fp + fn + tp):.2f}")

print(f"Precision: {tp / (tp + fp):.2f}")

print(f"Recall: {tp / (tp + fn):.2f}")

print(f"F1-Score: {2 * (precision * recall) / (precision + recall):.2f}")
```

Confusion Matrix:

[[4 0]

[0 2]]

Predicted value for age 35: [0]

True Negatives (TN): 4

False Positives (FP): 0

False Negatives (FN): 0

True Positives (TP): 2

TASK NO 3:

ROC (Receiver Operating Characteristic) Curve

What is an ROC Curve?

The ROC curve is a graphical representation that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. It's a plot of the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings.

- **True Positive Rate (TPR)**: Also known as Recall or Sensitivity, it measures the proportion of actual positives that are correctly identified by the model. TPR=TPTP+FN\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}TPR=TP+FNTP
- **False Positive Rate (FPR)**: It measures the proportion of actual negatives that are incorrectly identified as positives by the model. FPR=FPFP+TN\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}FPR=FP+TNFP

Understanding the ROC Curve

- **X-axis (FPR)**: The ROC curve's x-axis represents the False Positive Rate. It ranges from 0 to 1, where 0 indicates no false positives and 1 indicates all negatives are falsely identified as positives.
- **Y-axis (TPR)**: The y-axis represents the True Positive Rate. It also ranges from 0 to 1, where 0 indicates no true positives and 1 indicates all actual positives are correctly identified.
- **The Diagonal Line**: A diagonal line from (0,0) to (1,1) represents a random classifier. A model whose ROC curve lies on this line is no better than random guessing.
- **The Curve**: The ROC curve for a good classifier will bow towards the top-left corner, indicating a high True Positive Rate and a low False Positive Rate.

AUC (Area Under the ROC Curve)

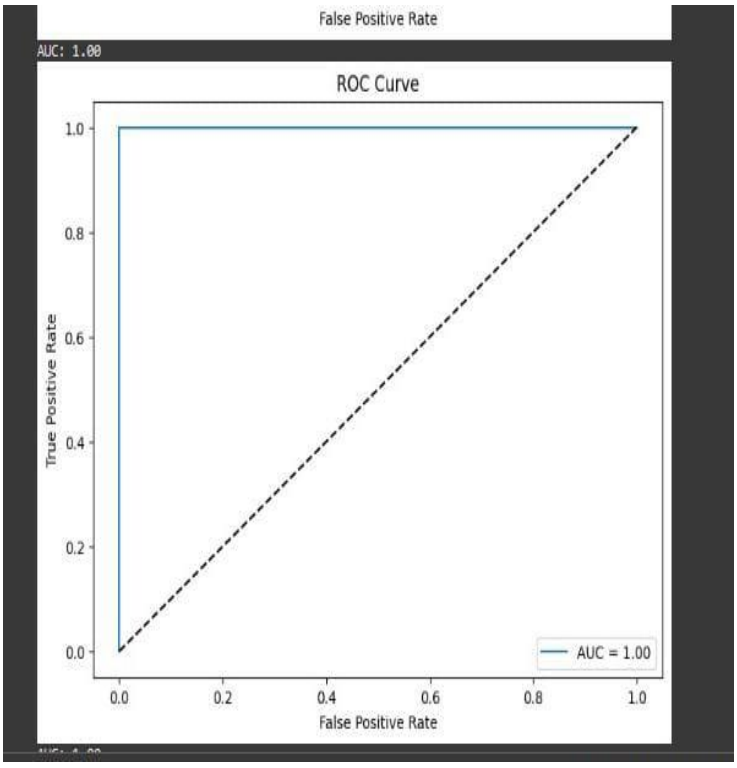
What is AUC?

AUC represents the area under the ROC curve and provides a single scalar value to assess the model's performance.

- **AUC Value**: The AUC value ranges from 0 to 1.
  - **AUC = 0.5**: This means the model is performing no better than random guessing.
  - **AUC < 0.5**: This indicates the model is performing worse than random guessing.
  - **AUC > 0.5**: This indicates the model is performing better than random guessing.
  - **AUC = 1**: This represents a perfect model.

- **Model Comparison**: AUC is useful for comparing the performance of different models. A higher AUC indicates a better performing model.
- **Threshold Independence**: AUC is not dependent on the chosen threshold. It evaluates the model's ability to distinguish between positive and negative classes over all possible thresholds.

```
# ROC curve calculate karen

fpr, tpr, thresholds = roc_curve(y_test, y_probs)

# AUC calculate karen

 auc = roc_auc_score(y_test, y_probs)

# ROC curve plot karen

 plt.figure(figsize=(8, 6)) plt.plot(fpr, tpr, label=f'AUC = {auc:.2f}')

plt.plot([0, 1], [0, 1], 'k--') #for diagonal line

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve')

plt.legend(loc='best')

plt.show()

print(f"AUC: {auc:.2f}")
```



TASK NO 4:

Explanation

1. **Performing Cross-Validation (`cross_val_score`)**:
   - `cross_val_score` function from `sklearn.model_selection` module is used to perform k-fold cross-validation.
   - `model`: This is your trained logistic regression model.
   - `df[['age']]`: This represents the feature data (age) used for training.
   - `df.bought_insurance`: This is the target data (whether insurance was bought or not).
   - `cv=k`: Specifies the number of folds (k) for cross-validation.
   - `scoring='accuracy'`: Indicates that we want to measure accuracy as the evaluation metric.
2. **Calculating Mean and Standard Deviation**:
   - `np.mean(cv_scores)`: Computes the mean (average) of the accuracy scores obtained from cross-validation.
   - `np.std(cv_scores)`: Computes the standard deviation of the accuracy scores, which measures the variability or spread of the accuracy values.
3. **Printing the Results**:
   - `print("Mean Cross-Validation Accuracy:", round(mean_cv_accuracy, 2))`: Prints the mean cross-validation accuracy rounded to two decimal places.
   - `print("Standard Deviation of Cross-Validation Accuracy:", round(std_cv_accuracy, 2))`: Prints the standard deviation of cross-validation accuracy rounded to two decimal places.

Why Cross-Validation is Important in Model Evaluation

- **Robust Performance Estimation**: Cross-validation provides a more reliable estimate of model performance compared to a single train-test split. It uses multiple train-test splits and averages the results, reducing the risk of overfitting or underfitting to a particular subset of data.
- **Varied Training and Testing Sets**: Each fold in cross-validation serves as both a training set and a validation set, ensuring that the model is evaluated on different combinations of data. This helps in understanding how the model generalizes to unseen data.
- **Model Selection**: Cross-validation helps in comparing different models or tuning hyperparameters by providing more accurate performance metrics across different subsets of data.
- **Mitigating Data Imbalance**: In cases of imbalanced datasets, cross-validation ensures that each class has an equal representation across different folds, leading to a more balanced evaluation of the model's performance

```
# Define the number of folds

k = 5



# Perform k-fold cross-validation

cv_scores = cross_val_score(model, df[['age']], df.bought_insurance, cv=k, scoring='accuracy')

# Calculate mean and standard deviation of the accuracy scores

mean_cv_accuracy = np.mean(cv_scores)

std_cv_accuracy = np.std(cv_scores)

# Print the results in multiple lines

print(f"{k}-Fold CV Accuracy:")

print(f"Mean = {mean_cv_accuracy:.2f}")

print(f"Standard Deviation = {std_cv_accuracy:.2f}")
```

5-Fold CV Accuracy:

Mean = 0.85

Standard Deviation = 0.15

Google Colab link

https://colab.research.google.com/drive/1oY4TIdpxGldLpElrqk5ZqTMGXbVtknK2?usp=sharing