Final Laboratory Assessment #3 – Linear Regression

Implementing Logistic Regression

Objective

In this lab activity, you will implement a logistic regression model to predict binary outcomes (whether a passenger survived the Titanic disaster) using a popular dataset. You will preprocess the data, train a logistic regression model, evaluate it, and score a separate dataset. Use CRISP-DM as your methodology.

Tools Required:

- **Google Colab** (for running the code)
- Python libraries: pandas, numpy, matplotlib, seaborn, scikit-learn

---

Datasets:

1. **Training Dataset**
    - **Source:** Titanic Dataset from Kaggle
    - Download the data set provided.
2. **Scoring Dataset**
    - A small dataset for scoring predictions (score_data.csv), which is also provided.

Instructions

1. Setting Up the Environment in Google Colab

- Open Google Colab and create a new notebook.
- Upload both the Titanic training dataset (train.csv) and the scoring dataset (score_data.csv) into the notebook.

2. Data Preprocessing

- **Load the dataset** using pandas:

```
import pandas as pd
train_data = pd.read_csv('train.csv')
```

- **Explore the data**:
    - Check for missing values.
    - Use train_data.info() and train_data.describe() to understand the dataset.
    - Visualize key features like Age, Fare, and Survived using seaborn:

```
import seaborn as sns
sns.countplot(x='Survived', data=train_data)
```

- **Handle missing data**:
  - Impute missing values or drop rows with missing values using train_data.dropna() or train_data.fillna().
  - For columns like Age, you may choose to impute using the mean or median.
- **Encode categorical variables**:
  - Use one-hot encoding for categorical features like Sex and Embarked:

    ```
    train_data = pd.get_dummies(train_data, columns=['Sex', 'Embarked'])
    ```
- **Split the dataset**:
  - Split the data into features (X) and target (y):

    ```
    X = train_data.drop(columns=['Survived', 'Name', 'Ticket', 'Cabin'])
    y = train_data['Survived']
    ```
  - Split the data into training and test sets using train_test_split:

    ```
    from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    ```

3. Model Training with Logistic Regression

- **Import the Logistic Regression model** from scikit-learn and train it on the training data:

  ```
  from sklearn.linear_model import LogisticRegression
  model = LogisticRegression()
  model.fit(X_train, y_train)
  ```
- **Evaluate the model**:
  - Use the trained model to make predictions on the test set:

    ```
    y_pred = model.predict(X_test)
    ```
  - Evaluate the model performance using metrics such as accuracy, precision, recall, F1 score, and ROC-AUC:

    ```
    from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
    print("Accuracy:", accuracy_score(y_test, y_pred))
    print("Confusion Matrix:")
    print(confusion_matrix(y_test, y_pred))
    print("Classification Report:")
    print(classification_report(y_test, y_pred))
    ```
  - You can also visualize the confusion matrix and ROC curve:

    ```
    import matplotlib.pyplot as plt
    from sklearn.metrics import roc_curve, auc
    ```

```
fpr, tpr, thresholds = roc_curve(y_test,
model.predict_proba(X_test)[:,1])
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (area =
{roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc='lower right')
plt.show()
```

4. Scoring New Data

- **Load the scoring dataset** (score_data.csv) into a DataFrame:
  ```
  scoring_data = pd.read_csv('score_data.csv')
  ```
- **Preprocess the scoring data** to match the format of the training data:
  - Apply the same encoding to the Sex and Embarked columns:
    ```
    scoring_data = pd.get_dummies(scoring_data, columns=['Sex', 'Embarked'])
    ```
  - Ensure that all features in score_data.csv match the training data's feature set, and handle any missing values.
- **Make predictions** using the trained model:
  ```
  predictions = model.predict(scoring_data.drop(columns=['PassengerId']))
  scoring_data['Survived'] = predictions
  ```
- **Save the predictions** to a new CSV file:
  ```
  scoring_data[['PassengerId', 'Survived']].to_csv('predictions.csv',
  index=False)
  ```

5. Documentation

- Add comments throughout your code explaining each step.
- At the end of the notebook, summarize your findings, discuss the model's performance, and describe any challenges you encountered.

Rubric

Criteria	Exemplary (5 pts)	Proficient (4 pts)	Developing (3 pts)	Needs Improvement (1-2 pts)
Data Preprocessing	Complete and correct preprocessing with insights.	Minor issues but functional preprocessing.	Preprocessing attempted but with significant gaps.	Minimal or incorrect preprocessing.
Model Training	Logistic regression implemented correctly; thorough evaluation of metrics.	Logistic regression implemented; basic metric evaluation.	Logistic regression implemented with errors or poor evaluation.	Little to no effort in model implementation.
Scoring New Data	Scoring dataset processed and predictions saved accurately.	Scoring dataset processed with minor issues.	Scoring dataset attempted but incomplete or incorrect.	Scoring not attempted or severely flawed.
Code Quality and Documentation	Clear, concise, and well-documented code.	Code is functional but with limited documentation.	Code is functional but unclear or poorly organized.	Code is disorganized or largely uncommented.
Analysis and Insights	Thorough analysis with insights and visualizations.	Adequate analysis and visualizations.	Minimal analysis; few visualizations.	No meaningful analysis or visualizations.

Total Points: ___ / 25

Deliverables

1. Completed Google Colab notebook (.ipynb).
2. predictions.csv file containing results for the scoring dataset.
3. Brief report summarizing findings and challenges (within the notebook).

Deadline:

Submission Instructions:
Submit your Google Colab notebook link and the predictions.csv file via [insert submission platform].