Assignment No. 2

Problem Statement: Exploring data analysis by applying various preprocessing techniques and performing EDA for Linear Regression models.

Objective: To perform Exploratory Data Analysis (EDA) and preprocessing on a dataset to prepare it for Linear Regression modeling. The process includes handling missing data, analyzing correlations, encoding categorical variables, feature scaling, and visualizing key patterns in the data to improve model accuracy.

Prerequisite:

1. A Python environment with essential libraries like pandas, numpy, matplotlib, seaborn, and scikit-learn.
2. Basic knowledge of Python, statistics, and machine learning principles.
3. Understanding of Linear Regression and its assumptions, such as linearity, normality, and absence of multicollinearity.

Theory: Linear Regression

- A statistical and machine learning approach used to model relationships between independent (features) and dependent (target) variables.
- The goal is to find the best-fitting line that minimizes the difference between actual and predicted values.

1. Types of Linear Regression

a) Simple Linear Regression

- Involves one independent variable (X) and one dependent variable (Y).
- Equation: $Y=\beta0+\beta1X+\epsilon$ Y = \beta_0 + \beta_1X + \epsilon
    - Where:
        - YY = Dependent variable
        - XX = Independent variable
        - $\beta0$ \beta_0 = Intercept
        - $\beta1$ \beta_1 = Slope coefficient
        - ϵ \epsilon = Error term

b) Multiple Linear Regression

- Involves two or more independent variables.
- Equation: $Y=\beta0+\beta1X1+\beta2X2+...+\betanXn+\epsilon$ Y = \beta_0 + \beta_1X_1 + \beta_2X_2 + ... + \beta_nX_n + \epsilon

- Where:
  - X1,X2,...,XnX_1, X_2, ..., X_n = Independent variables
  - β0,β1,...,βn\beta_0, \beta_1, ..., \beta_n = Regression coefficients
  - ϵ\epsilon = Error term

2. Assumptions of Linear Regression

a) Linearity

- The relationship between independent variables and the dependent variable should be linear.
- Checked using scatter plots and residual plots.
- If violated, apply polynomial regression or log transformation.

b) Independence

- Observations must be independent.
- Issues arise in time-series data and grouped survey data.
- Checked using residual analysis.
- Fix using lagged variables or alternative models like ARIMA.

c) Homoscedasticity (Constant Variance of Residuals)

- Residual variance should be constant across all values.
- Checked using residual vs. fitted value plots.
- Fix using log transformation or alternative models.

d) Normality of Residuals

- Residuals should follow a normal distribution.
- Checked using histograms and Q-Q plots.
- Fix using transformations like log or Box-Cox.

e) No Multicollinearity

- Independent variables should not be highly correlated.
- Checked using a correlation matrix and Variance Inflation Factor (VIF).
- Fix by dropping or combining variables, or using PCA.

3. Feature Selection in Linear Regression

- Correlation Analysis: Removes highly correlated features.
- Backward Elimination: Removes high p-value features.

- Forward Selection: Adds features that improve model performance.
- Lasso Regression: Shrinks coefficients to zero, eliminating less relevant features.

4. Performance Evaluation Metrics

a) Mean Absolute Error (MAE)

- Measures the average absolute difference between actual and predicted values.

b) Mean Squared Error (MSE)

- Penalizes larger errors more than MAE.

c) Root Mean Squared Error (RMSE)

- Square root of MSE, providing error values in the same unit as the dependent variable.

d) R-Squared (R^2)

- Represents the proportion of variance explained by independent variables.

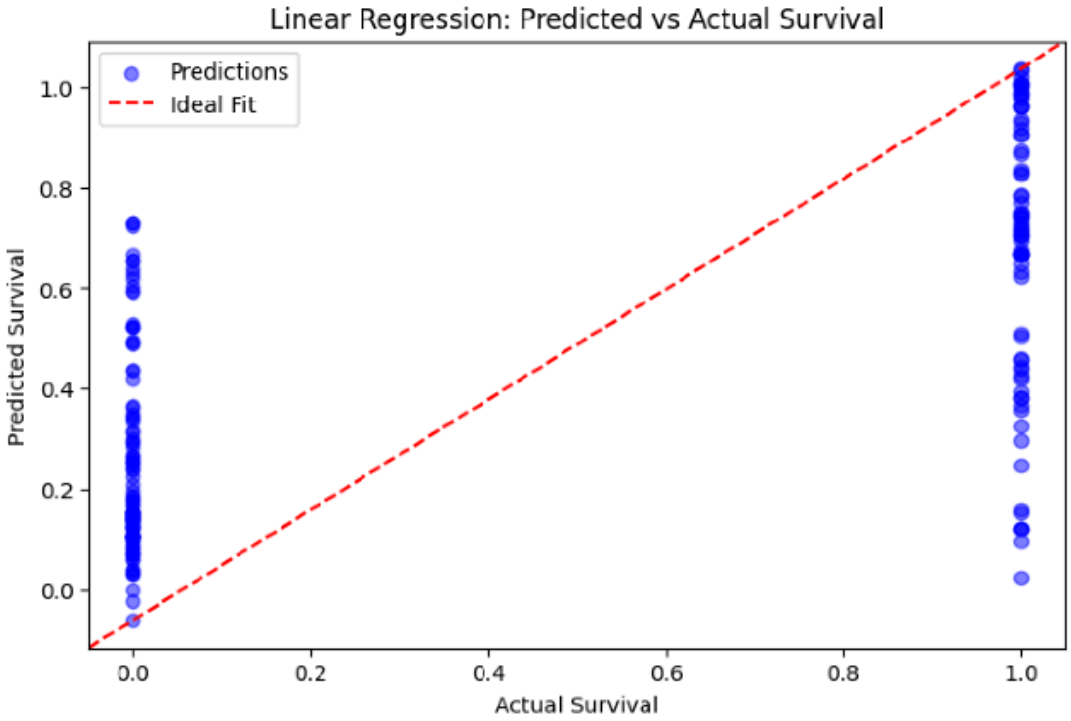
5. Practical Applications of Linear Regression

- **Business & Economics**: Sales and stock price forecasting.
- **Healthcare**: Predicting patient recovery time.
- **Marketing**: Forecasting customer demand.
- **Finance**: Credit risk assessment.
- **Real Estate**: Predicting house prices.

Code/Output:

```python
#Multiple Linear regression
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import numpy as np

# Select Features and Target
X = df[['Pclass', 'Age', 'SibSp', 'Parch', 'Fare', 'Sex_male', 'Embarked_Q', 'Embarked_S']]
y = df['Survived']

# Split dataset (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Linear Regression Model
model = LinearRegression()
model.fit(X_train, y_train)

# Predictions
y_train_pred = model.predict(X_train)
y_test_pred = model.predict(X_test)

# Model Evaluation
train_mse = mean_squared_error(y_train, y_train_pred)
train_rmse = np.sqrt(train_mse)
test_mse = mean_squared_error(y_test, y_test_pred)
test_rmse = np.sqrt(test_mse)

print("Training MSE:", train_mse)
print("Training RMSE:", train_rmse)
print("Testing MSE:", test_mse)
print("Testing RMSE:", test_rmse)
print("Model Coefficients:", model.coef_)
print("Model Intercept:", model.intercept_)

plt.figure(figsize=(8,5))
plt.scatter(y_test, y_test_pred, alpha=0.5, color="blue", label="Predictions")
plt.plot([0, 1], [0, 1], transform=plt.gca().transAxes, color="red", linestyle="--", label="Ideal Fit")
plt.xlabel("Actual Survival")
plt.ylabel("Predicted Survival")
plt.title("Linear Regression: Predicted vs Actual Survival")
plt.legend()
plt.show()
```

```
Training MSE: 0.14460581250588436
Training RMSE: 0.3802707095029597
Testing MSE: 0.135074012314622
Testing RMSE: 0.3675241656199249
Model Coefficients: [-0.15450237 -0.06103188 -0.03885891 -0.01957555  0.00823382 -0.51402058
 -0.02452473 -0.07141985]
Model Intercept: 1.156592483619219
```



Conclusion:

This assignment explored the fundamental concepts of Linear Regression, including its types, assumptions, and preprocessing techniques essential for effective model building. We performed data analysis by handling missing values, encoding categorical features, and scaling numerical data to meet Linear Regression assumptions. The importance of detecting and addressing multicollinearity, normality, and homoscedasticity was also highlighted. Finally, performance evaluation metrics such as MAE, MSE, RMSE, and R^2 were used to assess the model's accuracy. The insights gained from this study help improve model interpretability and predictive performance in real-world applications.