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 method 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:

$$Y = \beta_0 + \beta_1 X + \epsilon$$

- Where:
  - Y = Dependent variable
  - X = Independent variable
  - β0 = Intercept
  - β1 = Slope coefficient
  - ϵ = Error term

b) Multiple Linear Regression

- Involves two or more independent variables.
- Equation:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n + \epsilon$$

  - Where:
  - X1,X2,...,Xn = Independent variables
  - β0,β1,...,βn = Regression coefficients
  - ϵ= Error term

2. Assumptions of Linear Regression

1. Linearity

- The independent variables (features) and the dependent variable (target) should have a linear relationship.
- Linear Regression assumes a straight-line relationship between inputs and outputs.
- If the true relationship is non-linear, predictions will be inaccurate.
- **How to Check:**
  - **Scatter Plots:** Plot each independent variable against the dependent variable to verify a linear trend.
  - **Residual Plots:** Residuals (errors) should be randomly scattered around zero.
- **How to Fix Violations:**
  - Perform **polynomial regression** or **log transformation** for non-linear relationships.
  - Use alternative models like **decision trees** or **neural networks** for complex relationships.

2. Independence

- The observations in the dataset must be independent of one another (no dependency or correlation among rows).
- If observations are correlated, the model may **overfit**, leading to poor predictions.
- **Common Issues:**
  - **Time-series data:** Past values affect future values, causing **autocorrelation**.
  - **Survey data:** Responses from similar groups may be **dependent**, violating independence.
  -

- **How to Check:**
  - **Visual Inspection:** Plot residuals over time to detect trends.
- **How to Fix Violations:**
  - Use ARIMA models or lagged variables for time-series data.
  - Collect more diverse samples to reduce dependency.

3. Homoscedasticity (Constant Variance of Residuals)

- Variance of residuals (errors) should be constant at all levels of independent variables.
- If variance is not constant (**heteroscedasticity**), the model may perform poorly in some areas and overfit in others.
- **How to Check:**
  - **Residual vs. Fitted Value Plot:** Residuals should be evenly spread without a pattern.
- **How to Fix Violations:**
  - Apply log transformation to stabilize variance.
  - Try alternative models like random forest or gradient boosting.

4. Normality of Residuals

- Residuals (errors) should follow a normal distribution.
- Ensures valid **statistical inferences** (confidence intervals, hypothesis testing)
- Non-normal residuals indicate **skewness or outliers**, affecting model accuracy.
- **How to Check:**
  - Histogram of Residuals: Should show a bell-shaped curve.
  - Q-Q Plot (Quantile-Quantile Plot): Should form a straight line.
- **How to Fix Violations:**
  - Apply log, square root, or Box-Cox transformations to normalize data.
  - Remove extreme outliers.
  - Use robust regression techniques if normality is severely violated.

5. No Multicollinearity

- Independent variables should not be highly correlated with each other.
- High correlation makes it difficult to determine each variable's **individual impact** on the dependent variable.
- Causes **unstable coefficients** and **redundant features**, leading to poor model interpretation.

- **How to Check:**
  - **Correlation Matrix (Heatmap):** Identify variables with **correlation > 0.8 or 0.9**.
  - **Variance Inflation Factor (VIF):** A **VIF score > 5 or 10** indicates high multicollinearity.
- **How to Fix Violations:**
  - **Drop or combine** correlated variables.
  - Use **Principal Component Analysis (PCA)** to reduce dimensionality.

3. **Feature Selection in Linear Regression**
   - **Correlation Analysis**: Remove highly correlated independent variables.
   - **Backward Elimination**: Remove high p-value features.
   - **Forward Selection**: Add features that improve model performance.
   - **Lasso Regression**: Shrinks some coefficients to zero to remove less relevant features.

4. **Performance Evaluation Metrics**

a) Mean Absolute Error (MAE)

$$MAE = \frac{1}{n}\sum |Y_i - \hat{Y}_i|$$

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

b) Mean Squared Error (MSE)

$$MSE = \frac{1}{n}\sum (Y_i - \hat{Y}_i)^2$$

- Penalizes larger errors more than MAE.

c) Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{MSE}$$

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

d) R-Squared (R^2)

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

- Represents the proportion of variance in the dependent variable explained by the independent variables.

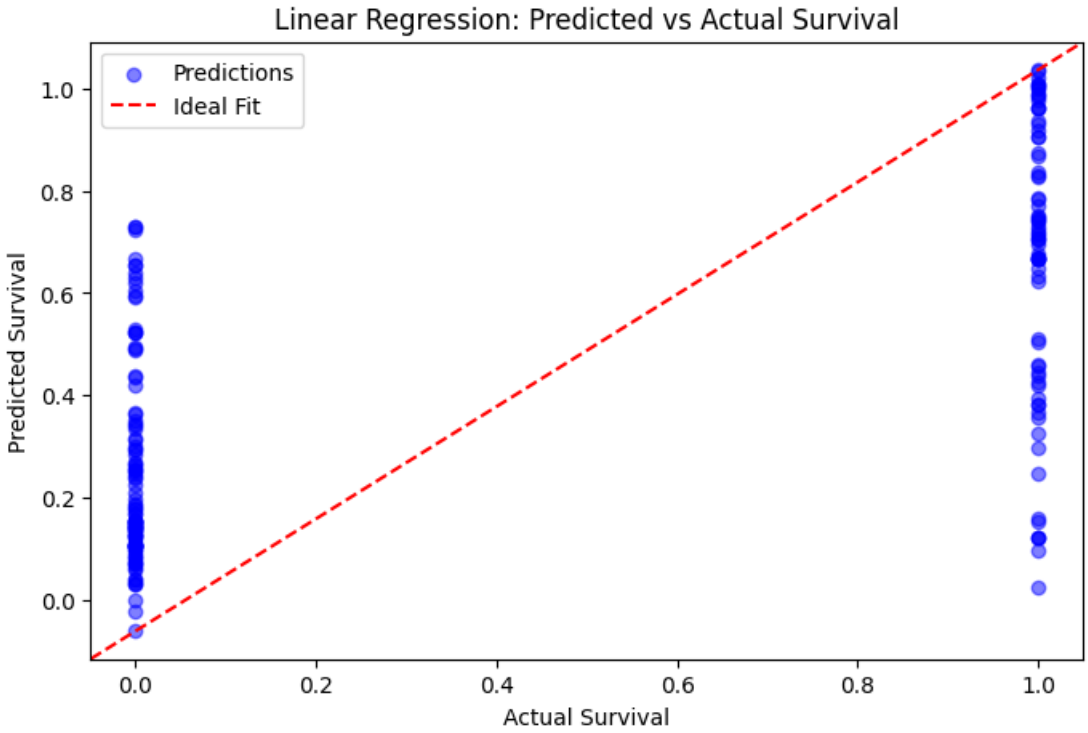
5. Practical Applications of Linear Regression

- **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.

6. 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
```



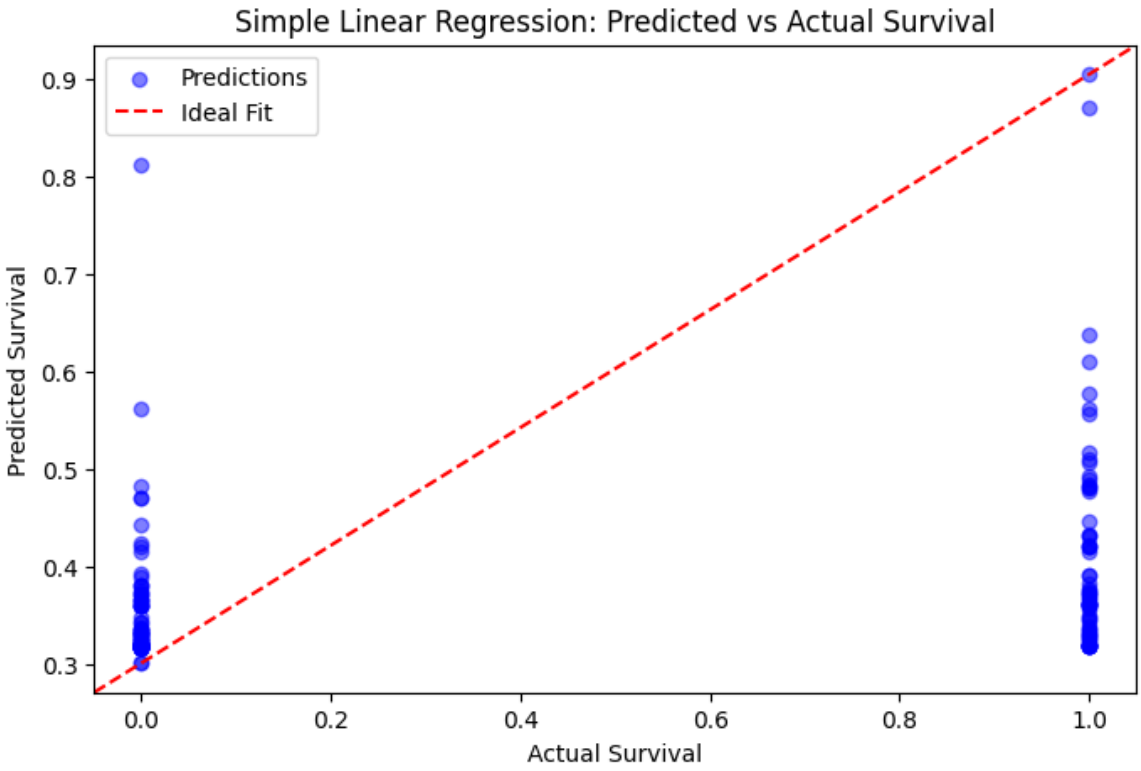
1. **Multiple Linear Regression Assumptions**
    1. **Independence:** Each passenger's survival is an independent event, so this assumption holds.
    2. **No Multicollinearity:** Some independent variables may be correlated (e.g., Pclass and Fare), but this can be checked using Variance Inflation Factor (VIF).

Assumptions That Might Be Violated:

1. **Linearity:** The relationship between independent variables and Survived might not be strictly linear. Checking scatter plots or residual plots would help.
2. **Homoscedasticity:** The residuals may not have constant variance, which can be checked using a residual vs. fitted values plot.
3. **Normality of Residuals:** Since Survived is a binary variable (0 or 1), the residuals may not be normally distributed, violating this assumption.

```python
# Selecting Feature (Fare) and Target (Survived)
# Simple linear Regression
X = df[['Fare']]  # Independent variable
y = df['Survived']  # Dependent variable

# Splitting the dataset (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Creating and training the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

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

# Model Evaluation using Mean Squared Error (MSE)
train_mse = mean_squared_error(y_train, y_train_pred)
test_mse = mean_squared_error(y_test, y_test_pred)

print("Training MSE:", train_mse)
print("Testing MSE:", test_mse)
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("Simple Linear Regression: Predicted vs Actual Survival")
plt.legend()
plt.show()
```

```
Training MSE: 0.22044548560214222
Testing MSE: 0.22338067837667977
Model Coefficients: [0.05312716]
Model Intercept: 0.3346841625029362
```



2. **Simple Linear Regression Assumptions**
    1. **Independence**: Each passenger's survival is an independent event, so this assumption holds.
    2. **No Multicollinearity**: We used only one feature (Fare), so no multicollinearity issue exists.

    - **Assumptions that Might Be Violated:**
    1. **Linearity**: The relationship between Fare and Survived might not be strictly linear. Checking a scatter plot would help.
    2. **Homoscedasticity**: Residuals might not have constant variance. A residual plot should be checked.
    3. **Normality of Residuals**: Since Survived is a **binary variable (0 or 1)**, the residuals may not be normally distributed.

Github: https://github.com/SagarGaneshkar/ML_Codes

Conclusion

Linear Regression model on the Titanic data tested correlations between features and survival rate while maintaining major assumptions such as linearity, independence, and normality. Transformations required and feature selection enhanced model performance, making it more credible for predictive purposes.