

Aim: Perform Regression Analysis using Scipy and Sci-kit learn.

Theory:

Regression analysis is a statistical technique used to model and analyze relationships between variables. It is commonly used for predicting numerical outcomes and identifying trends. There are different types of regression, with Linear Regression being the most fundamental.

Regression analysis helps in understanding:

- The relationship between dependent and independent variables.
- The impact of independent variables on the dependent variable.
- Predicting values based on trends in data.

Mathematically, it follows the equation:

$$y = \beta_0 + \beta_1 x + \epsilon$$

Where:

y is the dependent variable (what we predict).

x is the independent variable (the predictor).

β_0 is the intercept (constant term).

β_1 is the coefficient (slope).

ϵ is the error term (random noise).

There can also exist more than one independent variable, in such a case, regression follows the equation:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n$$

Types of Regression:

- Linear Regression (Simple & Multiple)
- Polynomial Regression (Non-linear relationship)
- Logistic Regression (Classification problems)
- Ridge & Lasso Regression (Regularization techniques)
- Support Vector Regression (SVR)
- Decision Tree & Random Forest Regression

Steps:

a. Linear Regression

1. Load the dataset

```
# Load dataset
df = pd.read_csv("/content/train.csv")
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103904 entries, 0 to 103903
Data columns (total 23 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   Gender                                     103904 non-null object
1   Customer Type                             103904 non-null object
2   Age                                         103904 non-null int64
3   Type of Travel                             103904 non-null object
4   Class                                       103904 non-null object
5   Flight Distance                           103904 non-null int64
6   Inflight wifi service                     103904 non-null int64
7   Departure/Arrival time convenient         103904 non-null int64
8   Ease of Online booking                    103904 non-null int64
9   Gate location                             103904 non-null int64
10  Food and drink                            103904 non-null int64
11  Online boarding                           103904 non-null int64
12  Seat comfort                              103904 non-null int64
13  Inflight entertainment                    103904 non-null int64
14  On-board service                          103904 non-null int64
15  Leg room service                          103904 non-null int64
16  Baggage handling                          103904 non-null int64
17  Checkin service                           103904 non-null int64
18  Inflight service                           103904 non-null int64
19  Cleanliness                               103904 non-null int64
20  Departure Delay in Minutes                103904 non-null int64
21  Arrival Delay in Minutes                  103904 non-null float64
22  satisfaction                               103904 non-null int64
dtypes: float64(1), int64(18), object(4)
memory usage: 18.2+ MB
```

The dataset consists of 103,904 entries with 23 columns, capturing various aspects of airline passengers' experiences and satisfaction levels. It contains a mix of categorical and numerical variables. The **categorical** attributes include **Gender**, **Customer Type**, **Type of Travel**, **Class**, and **Satisfaction**, which provide insights into passenger demographics and travel preferences. The **numerical** attributes include **Age**, **Flight Distance**, **Departure and Arrival Delays**, and various in-flight service ratings such as WiFi service, food and drink, seat comfort, and baggage handling, all measured on an integer scale. Arrival delay will act as our target variable for the purpose of performing regression.

2. Perform all the necessary preprocessing steps

- a. Handling missing values
- b. Drop unnecessary columns
- c. Data transformation

```
le = LabelEncoder()
df["satisfaction"] = le.fit_transform(df["satisfaction"])
```

Convert categorical columns to binary data.

3. Split data and train the model

```
# Split data for Linear Regression
X_linear = df[selected_features_linear]
y_linear = df["Arrival Delay in Minutes"]
X_train_lin, X_test_lin, y_train_lin, y_test_lin = train_test_split(X_linear, y_linear, test_size=0.2, random_state=42)
```

Split data into 80% training and 20% testing datasets.

```
# Standardize features for Linear Regression
scaler = StandardScaler()
X_train_lin = scaler.fit_transform(X_train_lin)
X_test_lin = scaler.transform(X_test_lin)

# Train Linear Regression Model
linear_model = LinearRegression()
linear_model.fit(X_train_lin, y_train_lin)
y_pred_lin = linear_model.predict(X_test_lin)
```

Regression is sensitive to sudden changes in ranges of independent variables, to avoid this we normalize our numerical columns. We use the z score transformation.

Then we train our model, and use the model to predict the testing dataset.

4. Evaluation

```
# Evaluate Linear Regression
mse_lin = mean_squared_error(y_test_lin, y_pred_lin)
accuracy_lin = 1 - (mse_lin / np.var(y_test_lin)) # R-squared equivalent
print(f"Accuracy (Linear Regression): {accuracy_lin*100:.2f}%")
print("Mean square value : " , mse_lin)

coefficients = linear_model.coef_
intercept = linear_model.intercept_

# Format the equation
equation = "y = " + " + ".join([f"{coeff:.4f}*{feature}" for coeff, feature in zip(coefficients, selected_features_linear)])
equation += f" + {intercept:.4f}"

print("Regression Equation:")
print(equation)
```

We evaluate a linear regression model by calculating the Mean Squared Error (MSE) and accuracy to measure performance. It then extracts the model's coefficients and intercept to construct a regression equation.

```
Accuracy (Linear Regression): 91.75%
Mean square value : 115.98955069314997
```

Regression Equation:

$$y = -0.1475 * \text{Flight Distance} + 37.3971 * \text{Departure Delay in Minutes} + -0.0577 * \text{Inflight wifi service} + -0.0512 * \text{Seat comfort}$$

The regression model achieves **91.75% accuracy**, indicating a strong fit. The regression equation suggests that Departure Delay in Minutes has the highest impact on the dependent variable, followed by On-board service, Inflight WiFi service, and Seat comfort, while Flight Distance has a small negative effect.

The mean squared error (MSE) of **115.99** suggests some variance in predictions, but overall, the model performs well in explaining the relationship between these factors and the target variable.

b. Logistic regression :

It's a supervised classification algorithm used to predict binary (0/1) or multi-class outcomes. Despite its name, it is actually a classification model, not a regression model.

- Logistic Regression estimates the probability $P(Y=1 | X)$ using the **sigmoid (logistic) function**. The output is a probability value between **0 and 1**, which is then classified using a **threshold** (e.g., ≥ 0.5 as class 1, < 0.5 as class 0).

1. Selecting the required feature through feature selection basically uses RFE (Recursive Feature Elimination) with Logistic Regression to select the 10 most important features.

```
# Selecting all numerical features for Logistic Regression
selected_features_logistic = df.select_dtypes(include=[np.number]).columns.tolist()
selected_features_logistic.remove("satisfaction")

# Feature Selection using RFE for Logistic Regression
X_logistic = df[selected_features_logistic]
y_logistic = df["satisfaction"]
logistic_selector = RFE(LogisticRegression(max_iter=5000, solver="lbfgs"), n_features_to_select=10)
logistic_selector.fit(X_logistic, y_logistic)
selected_features_logistic = X_logistic.columns[logistic_selector.support_].tolist()
print(selected_features_logistic)
```

2. Splits the selected feature dataset into training (80%) and testing (20%) set

```
# Split data for Logistic Regression
X_logistic = df[selected_features_logistic]
X_train_log, X_test_log, y_train_log, y_test_log = train_test_split(X_logistic, y_logistic, test_size=0.2, random_state=42, stratify=y_logistic)
```

3. Removing the outliers to enhance the model and which scales data while reducing the impact of outliers.

```
# Use RobustScaler for better handling of outliers
scaler_log = RobustScaler()
X_train_log = scaler_log.fit_transform(X_train_log)
X_test_log = scaler_log.transform(X_test_log)
```

4. Training and evaluating the model

```
logistic_model = LogisticRegression(max_iter=5000, solver="lbfgs", C=0.5, class_weight="balanced")
logistic_model.fit(X_train_log, y_train_log)
y_pred_log = logistic_model.predict(X_test_log)

# Evaluate Logistic Regression
accuracy_log = accuracy_score(y_test_log, y_pred_log)
f1_log = f1_score(y_test_log, y_pred_log, average="weighted")
conf_matrix_log = confusion_matrix(y_test_log, y_pred_log)
classification_report_log = classification_report(y_test_log, y_pred_log)

print(f"Accuracy : {accuracy_log * 100:.2f}%")
print("F1 Score :", f1_log)
print("Confusion Matrix:\n", conf_matrix_log)
```

The Logistic Regression model is trained with hyperparameter tuning (max_iter=5000, solver="lbfgs", C=0.5, and class_weight="balanced"), ensuring better convergence and handling of imbalanced data. Finally, the model's performance is evaluated using accuracy, F1 score, confusion matrix, and classification report, which help analyze its effectiveness in predicting the target variable.

```
Accuracy : 80.60%
F1 Score : 0.8065822767400087
Confusion Matrix:
[[9450 2326]
 [1706 7299]]
```

The output shows the selected features used for training the Logistic Regression model, including aspects like "Inflight Wifi Service," "Seat Comfort," and "Online Boarding." The model achieved an **accuracy** of **80.60%** and an F1 score of 0.80, indicating a good balance between precision and recall. The confusion matrix reveals that the model correctly classified **9450 instances as negative and 7299 as positive, while misclassifying 2392 false positives and 1706 false negatives.**

Conclusion:

The Linear Regression model achieved 91.75% accuracy, highlighting a strong relationship between Departure Delay, On-board Service, Inflight WiFi Service, and Seat Comfort with the target variable. Despite an MSE of 115.99, the model provided a reasonably good fit. For classification, Logistic Regression with Recursive Feature Elimination (RFE) was applied, resulting in an 80.60% accuracy and an F1 score of 0.80, indicating balanced classification performance. The confusion matrix further validated the model's effectiveness in distinguishing between classes.

In this we were able to understand how not only by selecting but to use feature engineering is utmost necessary for your model to be working really properly and accurate on unknown data