EXPERIMENT - 5

Aim: Regression Analysis
a. Perform Logistic Regression to find out relation between variables.
b. Apply regression Model techniques to predict the data on above dataset

Theory:

1. Regression Analysis

Regression analysis is a statistical method used to model the relationship between a dependent variable (target) and one or more independent variables (features). It helps in predicting outcomes and understanding how changes in predictors affect the target variable.

Regression can be classified into:

- **Linear Regression** (for continuous target variables)
- **Logistic Regression** (for categorical target variables)
- **Polynomial & Non-Linear Regression**
- **Regularized Regression (Lasso, Ridge, ElasticNet)**

2. Logistic Regression

Definition:
Logistic Regression is a classification algorithm used when the dependent variable is categorical (e.g., 0 or 1, Yes or No). It estimates the probability that a given input belongs to a particular class using the **sigmoid function**:

$$P(Y=1|X)=1 / 1+e-(\beta 0+\beta 1X1+\beta 2X2+...+\beta nXn)$$

where P(Y=1|X) is the probability of belonging to class 1.

Key Points:

- If **P > 0.5**, classify as **1**, else **0**
- Logistic Regression works well for **binary classification**
- It is widely used for **spam detection, disease prediction, fraud detection, etc.**

3. Regression Model Techniques for Prediction

Regression models predict numerical values based on input variables. Some common regression techniques are:

(a) Linear Regression

Used when the target variable is continuous. It models the relationship as a straight line

(b) Polynomial Regression

Used when the relationship is **non-linear**. It transforms the features into higher-degree polynomials.

4. Confusion Matrix

A **confusion matrix** is a table used to evaluate classification models by showing true vs. predicted values. It has four components:

	Predicted No Fall	**Predicted Hair Fall**
Actual No Fall (TN)	True Negative (TN)	False Positive (FP)
Actual Hair Fall (TP)	False Negative (FN)	True Positive (TP)

- **True Positives (TP)**: Correctly predicted "hair fall" cases.
- **True Negatives (TN)**: Correctly predicted "no hair fall" cases.
- **False Positives (FP)**: Wrongly predicted "hair fall" when it’s not.
- **False Negatives (FN)**: Missed actual "hair fall" cases.

5. Classification Report Parameters

The classification_report(y_test, y_pred) in Scikit-learn provides key metrics to evaluate the performance of a classification model:

1. **Precision**: Measures the accuracy of positive predictions.

   Precision=TP/(TP+FP)
   - High precision means fewer false positives.
   - Example: If predicting "hair fall," precision tells us how many of the predicted "hair fall" cases are actually correct.

2. **Recall (Sensitivity or True Positive Rate)**: Measures how many actual positive cases were correctly identified.

   Recall=TP/(TP+FN)
   - High recall means fewer false negatives.
   - Example: If recall is low, the model is missing cases of "hair fall."

3. **F1-Score**: Harmonic mean of precision and recall. It balances the two metrics.

   F1-Score=2×(Precision×Recall/Precision+Recall)
   - A good F1-score means both precision and recall are balanced.

4. **Support**: Number of actual occurrences of each class in y_test.

6. What and why Scaling?

Scaling is a data preprocessing technique that transforms numerical features into a standard range. Many machine learning models, including Logistic Regression, SVM, and Neural Networks, perform better when features have similar scales.

- **Handles different feature ranges** → Some features may have very large values (e.g., salary in thousands) while others have small values (e.g., age in years). This difference can cause some features to dominate the model.

Example:

Before Scaling

Feature	Age	Salary
Person 1	25	40,000
Person 2	30	60,000

After Scaling

Feature	Age (Scaled)	Salary (Scaled)
Person 1	-1.0	-1.2
Person 2	0.0	-0.2

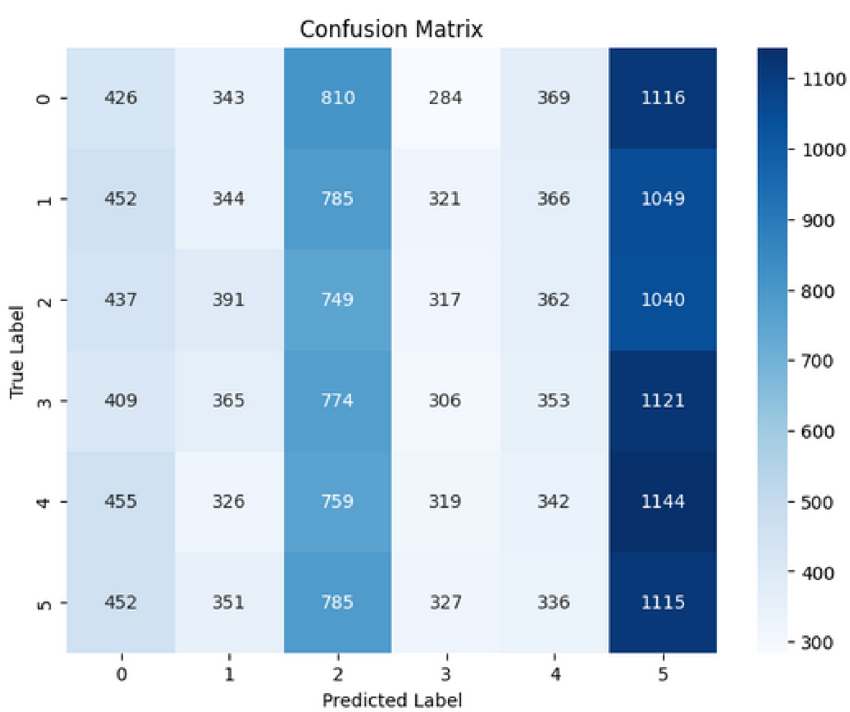
Program:

```
[1] import pandas as pd
    import numpy as np
    from sklearn.model_selection import train_test_split
    from sklearn.linear_model import LogisticRegression
    from sklearn.preprocessing import StandardScaler
    from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
    import matplotlib.pyplot as plt
    import seaborn as sns

    # Read the dataset
    df = pd.read_csv('hair_loss.csv')

    # Separate features and target
    X = df.drop('hair_fall', axis=1)
    y = df['hair_fall']

    # Split the data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Scale the features
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    # Train logistic regression model
    model = LogisticRegression(random_state=42, max_iter=1000)
    model.fit(X_train_scaled, y_train)
```

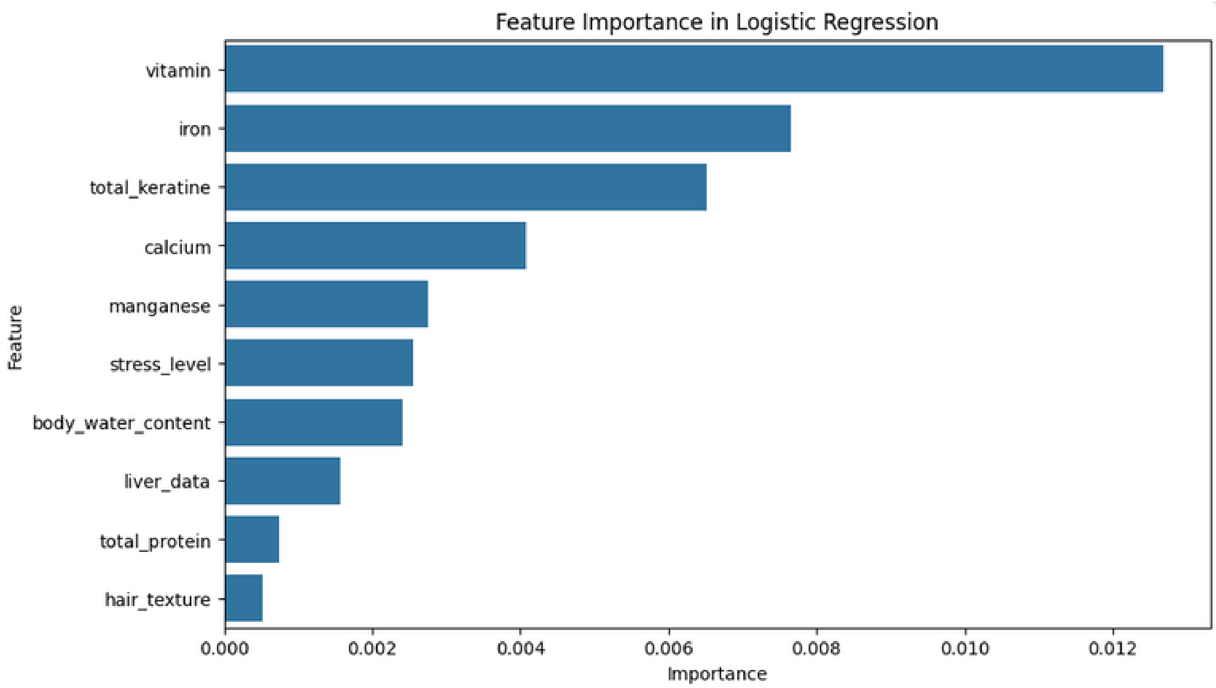
```
LogisticRegression
LogisticRegression(max_iter=1000, random_state=42)
```

```
[2] # Make predictions
    y_pred = model.predict(X_test_scaled)

    # Print model performance
    print("Accuracy:", accuracy_score(y_test, y_pred))
    print("\nClassification Report:")
    print(classification_report(y_test, y_pred))

    # Create confusion matrix
    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title('Confusion Matrix')
    plt.ylabel('True Label')
    plt.xlabel('Predicted Label')
    plt.show()
```

```
Accuracy: 0.1641

Classification Report:
              precision    recall  f1-score   support

           0       0.16      0.13      0.14      3348
           1       0.16      0.10      0.13      3317
           2       0.16      0.23      0.19      3296
           3       0.16      0.09      0.12      3328
           4       0.16      0.10      0.12      3345
           5       0.17      0.33      0.22      3366

    accuracy                           0.16     20000
   macro avg       0.16      0.16      0.15     20000
weighted avg       0.16      0.16      0.15     20000
```



```python
# Feature importance
feature_importance = pd.DataFrame({
    'Feature': X.columns,
    'Importance': abs(model.coef_[0])
})
feature_importance = feature_importance.sort_values('Importance', ascending=False)

# Plot feature importance
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feature_importance)
plt.title('Feature Importance in Logistic Regression')
plt.show()

# Make a sample prediction
sample_data = np.array([[312, 100, 1400, 249, 87, 55, 333, 44, 41, 368]])
sample_scaled = scaler.transform(sample_data)
prediction = model.predict(sample_scaled)
probability = model.predict_proba(sample_scaled)

print("\nSample Prediction:")
print("Predicted Class:", prediction[0])
print("Probability Distribution:", probability[0])
```




```
Sample Prediction:
Predicted Class: 4
Probability Distribution: [1.56042476e-01 9.94814672e-02 7.98466834e-02 2.51387412e-01
```

Conclusion: Thus, we have successfully performed Regression Analysis by finding out relation between variables and Applied regression Model techniques to predict the data on dataset.