Machine Learning Assignment-1

Overview

The project aims to create a model for predicate the load statues and the max load amount provided.

Using famous Machine learning technique (linear regression and logistic regression) and data virtualization and preparing

About Data

the data separate into two categories.

1- feature Data

Gender, Married, Dependents, Education, Income, Co-Applicant Income, Load Tenor, Credit History, Property Area.

2- Target Data

Max Load Amount, Load Status.

Loan_ID	Gender	Married	Dependents	Education	Income	Coapplicant_Income	Loan_Tenor	Credit_History	Property_Area	Max_Loan_Amount	Loan_Status
LP001002	Male	No	0	Graduate	5849	0	144	1	Urban		Y
LP001003	Male	Yes	1	Graduate	4583	1508	144	1	Rural	236.99	N
LP001005	Male	Yes	0	Graduate	3000	0	144	1	Urban	81.20	Y
LP001006	Male	Yes	0	Not Graduate	2583	2358	144	1	Urban	179.03	Y
LP001008	Male	No	0	Graduate	6000	0	144	1	Urban	232.40	Y
LP001011	Male	Yes	2	Graduate	5417	4196	144	1	Urban	414.50	Y
LP001013	Male	Yes	0	Not Graduate	2333	1516	144	1	Urban	123.99	Y
LP001014	Male	Yes	3+	Graduate	3036	2504	144	0	Semiurban	209.22	N
LP001018	Male	Yes	2	Graduate	4006	1526	144	1	Urban	208.81	Y
LP001020	Male	Yes	1	Graduate	12841	10968	144	1	Semiurban	449.00	N
LP001024	Male	Yes	2	Graduate	3200	700	144	1	Urban	126.56	Y
LP001027	Male	Yes	2	Graduate	2500	1840	144	1	Urban	148.74	Y
LP001028	Male	Yes	2	Graduate	3073	8106	144	1	Urban	363.42	Y
LP001029	Male	No	0	Graduate	1853	2840	144	1	Rural	166.53	N
LP001030	Male	Yes	2	Graduate	1299	1086	120	1	Urban	30.17	Y

the libraries use used:

```
import pandas as pd
import numpy as np
import copy, math
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.preprocessing import LabelEncoder
import seaborn as sns
```

Perform analysis & Preprocess.

First, we load the data to our environment.

then check if there is a missing data or not.

```python
# b(I)
def ismiss(missing_values):
    if missing_values:
        print("There are missing values in the Data.")
        return 1;
    else:
        print("There are no missing values in the Data.")
        return 0;
```

and clean up if there.

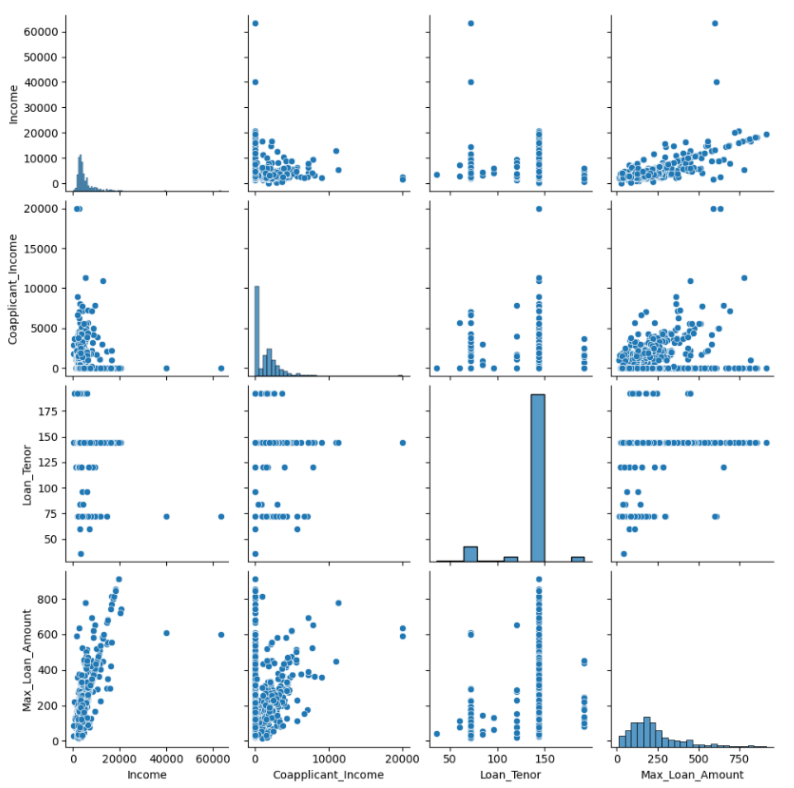
```python
#C(I)
missing_values = data.isnull().any().any()
if(ismiss(missing_values)==1):
    data.dropna(inplace=True)
```

and we did for loop check if the feature is a category or a numerical type and it it`s a category we encode it and if it`s a numerical we first find the scale of it and then do standardization on it.

```python
for column in data.columns:
    #B(II)
    if column == "Loan_ID" :
        continue
    if data[column].dtype == 'object':
        #C(IV , V)
        label_encoder = LabelEncoder()
        data[column] = label_encoder.fit_transform(data[column])
    else:
        if(data[column].nunique()==2):
            continue
        numerical_columns = np.append(numerical_columns,column)
        if(column == "Max_Loan_Amount"):
            continue
        #B(III)
        print(data[column].min() ,data[column].max())
        #C(VI)
        data[column] = zscore_normalize_features(data[column])
```

```python
#c(VI)
def zscore_normalize_features(X):
    mu     = np.mean(X, axis=0)
    sigma  = np.std(X, axis=0)
    X_norm = (X - mu) / sigma
    return X_norm
```

and then we did a pair plot between numerical columns w



and then we shuffled the data and split with percentage 80% into train and test data

```
data=np.array((data))
#C(III)
np.random.shuffle(data)
train_size = 0.8
train_data, test_data = train_test_split(data, train_size=train_size, random_state=42)
#C(II)
x_train, y_train = train_data[:, 1:-2], train_data[:, -2:]
x_test, y_test = test_data[:, 1:-2], test_data[:, -2:]
```

Linear regression model

we implemented linear regression using sklearn library, and this is the factors of the model.

[12.60828387, 2.62333085, 5.73226535, -15.94247464, 115.50884819, 67.06617541, 50.16626144, 6.64994346, -13.24311213] & 222.2779334755963

and the R2 Score were as follow.

0.7059963117251045

```
reg = LinearRegression().fit(x_train, y_train.T[0])
y_predicted= reg.predict(x_train)
w_regression = reg.coef_
b_regression = reg.intercept_
print(reg.coef_,reg.intercept_)
print(r2_score(y_train.T[0], y_predicted))
```

```
[ 12.60828387   2.62333085   5.73226535 -15.94247464 115.50884819
  67.06617541  50.16626144   6.64994346 -13.24311213] 222.2779334755963
0.7059963117251045
```

logistic regression model

we did logistic model from scratch and here all code components.

1- **f_wb ,** do a computation of the equation 1/1-e^-z that we will use it next at cost function.

```
def f_wb(z):
    g = 1/(1+np.exp(-z))
    return g
```

2- **cost function,** compute the cost of the model to show how good the model is

```
def compute_cost_logistic(X, y, w, b):
    m = X.shape[0]
    cost = 0.0
    for i in range(m):
        z = np.dot(X[i],w) + b
        f_wb_i = f_wb(z)
        cost = cost - y[i] * np.log(f_wb_i) - ( 1 - y[i]) * np.log(1-f_wb_i)
    cost = cost / m
    return cost
```

3- **w & b derivative,** compute the partial derivative of the model coefficients which use to build gradient descent for find best w & b

```
def compute_gradient_logistic(X, y, w, b):
    m,n = X.shape
    dj_dw = np.zeros((n,))
    dj_db = 0.

    for i in range(m):
        z_i = np.dot(X[i],w) + b
        f_wb_i = f_wb(z_i)
        err = f_wb_i - y[i]
        for j in range(n):
            dj_dw[j] = dj_dw[j] + err * X[i,j]
        dj_db = dj_db + err
    dj_dw = dj_dw/m
    dj_db = dj_db/m
    return dj_db, dj_dw
```

4- **gradient descent,** use for getting best model coefficient that make the cost as low as possible.

```python
def gradient_descent(X,y,w_in,b_in,alpha,iterations):
    J_history = []
    w = copy.deepcopy(w_in)
    b = b_in
    for i in range(iterations):
        dj_db, dj_dw = compute_gradient_logistic(X, y, w, b)
        w = w - alpha * dj_dw
        b = b - alpha * dj_db
        if i<100000:
            J_history.append( compute_cost_logistic(X, y, w, b) )
        if i% math.ceil(iterations / 10) == 0:
            print(f"Iteration {i:4d}: Cost {J_history[-1]}   ")
    return w, b, J_history
```

apply it on our data:

```python
w_temp = [ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0]
b_temp = 0
w_logistic , b_logistic , j_history = gradient_descent(x_train, y_train.T[1], w_temp, b_temp, 0.01,4000)
print(w_logistic , b_logistic)
```

```
Iteration    0: Cost 0.6912987338811534
Iteration  400: Cost 0.5914547989810134
Iteration  800: Cost 0.5729486914666865
Iteration 1200: Cost 0.5594270260732804
Iteration 1600: Cost 0.5488719481168476
Iteration 2000: Cost 0.5403509573232079
Iteration 2400: Cost 0.5333231927572983
Iteration 2800: Cost 0.5274339148889454
Iteration 3200: Cost 0.5224344358597777
Iteration 3600: Cost 0.5181432997089046
[-0.18070169  0.384919   -0.08109239 -0.43573011  0.1095229  -0.02416577
 -0.0143741   1.74162832 -0.03526205] -0.5471753221654987
```

accuracy

We have to Get the predicated target and compare it to the original one for find the accuracy of the model.

1- get Predicted data of logistic model

```python
def predicted_logistic(X,w,b):
    m = X.shape[0]
    y_predicted = np.zeros(m)
    for i in range (m):
        z = np.dot(X[i],w) + b
        y_predicted[i] = f_wb(z)
    return y_predicted
```

2- apply the function of accuracy

```python
def accuracy(y_hat_logistic , y_logistic ):
    m = y_logistic.shape[0]
    miss = 0 ;
    for i in range(m):
        if y_hat_logistic[i] >= 0.5 :
            if(y_logistic[i]==0):
                miss = miss + 1
        if y_hat_logistic[i] < 0.5:
            if(y_logistic[i]==1):
                miss = miss + 1
    print("the accuracy of logistic model" , ((m-miss)/m ) * 100)
```

3- test the accuracy on our data

```python
accuracy(y_hat,y_test.T[1])
```

```
the accuracy of logistic model 85.43689320388349
```

New Data

we applied same Perform analysis and Preprocess except what on the targets.

```
new_data = pd.read_csv("loan_new.csv")
missing_values = new_data.isnull().any().any()
ismiss(missing_values)
new_data.dropna(inplace=True)
missing_values = new_data.isnull().any().any()
ismiss(missing_values)
numerical_columns = np.array(())
for column in new_data.columns:
    if column == "Loan_ID" :
        continue
    if new_data[column].dtype == 'object':
        label_encoder = LabelEncoder()
        new_data[column] = label_encoder.fit_transform(new_data[column])
    else:
        print(new_data[column].min() ,new_data[column].max())
        new_data[column] = zscore_normalize_features(new_data[column])
        if(new_data[column].nunique()==2):
            continue
        numerical_columns = np.append(numerical_columns,column)
new_data=np.array((new_data))
x_train_new = new_data[:, 1:]
print((x_train_new.shape))
```

```
There are missing values in the Data.
There are no missing values in the Data.
0 72529
0 24000
12.0 192.0
0.0 1.0
(314, 9)
```

and then predicate the Max Amount and loan status and this a sample of the data:

```
y_hat_reg = predict_regression(x_train_new,w_regression,b_regression)
y_hat_log = predicted_logistic(x_train_new , w_logistic ,b_logistic)
print (y_hat_reg ,"\n", y_hat_log)
```

Max Loan Amount

```
[ 206.78167954  191.16505506  252.44128807  128.61672437  202.53538373
  109.37861865  166.44918441  306.45627357  189.50168957  124.10533211
  183.14035265  371.3564639   179.09043401  208.59816111  274.35918191
  199.63354097  519.87653962   52.25051671  149.70232157  -53.02054104
```

Loan Status

```
[0.6044843  0.56621328 0.55643462 0.38865379 0.46774583 0.41523112
0.00876909 0.61656217 0.38631435 0.38763791 0.56976986 0.01574259
0.55745983 0.48923875 0.55119516 0.54190198 0.60425873 0.49535074
0.45065499 0.3779846  0.43512625 0.59179948 0.00774375 0.60503585
```