Simple Linear Regression Experiment Report

Darryl Vas Prabhu
Department of Computer Science and Engineering
University at Buffalo, Buffalo, NY 14260
dvasprab@buffalo.edu

1 Simple Linear Regression

[1]Data is the key to solving a number of problems in the world. It's a well known fact even in a traditional sense, we learn only from experience and the same is true for machines as well. When enough data is provided to a machine with appropriate learning models deployed, machines can use these patterns in the models to solve new problems. Starting with the simple model called linear regression, the term 'linear' comes from being in line, and regression comes from the definition regression in statistics i.e. the measure of the relation between the mean value of one variable (e.g. output) and corresponding values of other variables. In linear regression, we predict the extent to which one variable changes with respect to changes in the other. For example: the yield of crops based on rainfall etc.

In simple linear regression, we predict the output of the dependant variable $\mathbf{y}$ based on some function of a single independent variable $\mathbf{x}$.The linear model's notation doesn't seem to be far from the general equation of line on the co-ordinate system : $y = \mathbf{w}^T\mathbf{x} + b$, where y represents the dependent variable, that is, what we are trying to understand/explain/predict. $\mathbf{x}$ represents the independent variable. The intercept/bias, b, represents the value of y when $\mathbf{x}$ equals zero. The regression coefficient, $\mathbf{w}$, represents the variation observed in y associated with the increase of one unit of $\mathbf{x}$.

The formula for simple linear regression is give below

$$p(y|x;) = N(y|w0 + wTx;).$$

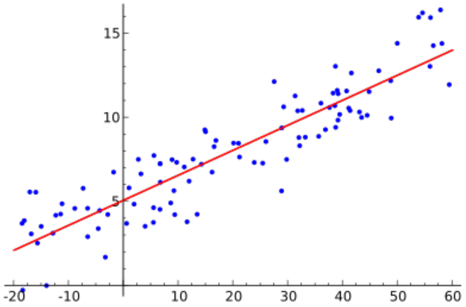


Figure 1: Simple Linear regression against data points

Figure 1[7] shows the linear regression line which is also called as the best fit line. The goal of linear regression is to find the line such that the distance between the data-points and the line is reduced. This distance is called as the residual distance and the goal is the achieve minimum residual distance. The data points can lie both above and below the best fit line. Hence while calculating the sum of the errors, it's possible that the result could be negative.This possibility is taken care using MSE(Mean-Square-Error). To understand simple linear regression model, one of the data sets from kaggle was used to predict the real estate price of houses per unit area. Although there are many parameters to this dataset, for our focus on simple linear regression, only the attribute "X3 distance to the nearest MRT station" was taken as the independent variable with "Y house price of unit area" as the dependant variable.

To train the simple regression model, we are given a the training data in terms of $\mathcal{D} \triangleq \{(\mathbf{x}_i, y_i)\}_{i=1}^N$. The least square loss is given as:

$$J(\mathbf{w}) = (xTx)^{-1}xTy.$$

[1] Simple Linear Regression.

To compute the model parameters we find the derivative of the squared loss with respect to the regression coefficient and normalize it to 0 to find the the regression coefficient.

$$\mathbf{w} = (\mathbf{X^T X})^{-1}\mathbf{X^T y}$$

2 Experiments

To illustrate the application of the simple regression an experiment was performed using data-set from [4] on price of house with respect to distance from MRT station'.Figure 2 Shows the negative slope indicating that as the distance of the house from MRT(Mass rapid Transport) increases , the price per unit area of the house decreases.

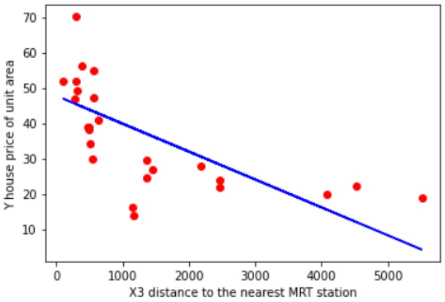
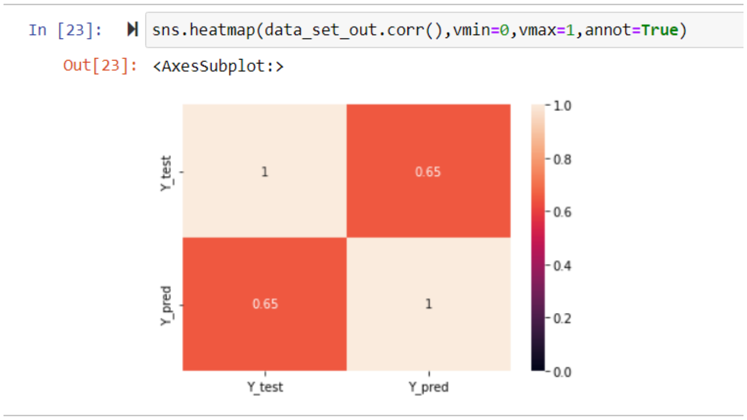


Figure 2: Data for linear regression.

Although the simple linear regression model has predicted the best fit line, we can see that most of the houses are within 1000 distance from the MRT station, resulting in the following regression line. If more linear data is present , the best fitted line would be further improved.

As noted from image Figure 3, the correlation between predicted values of price and the test prices is average. THe model is not completely accurate with respect to it's input owing to the lack of linearity in data of the independent variable which is the distance of the house from the MRT.



```
In [23]:  sns.heatmap(data_set_out.corr(),vmin=0,vmax=1,annot=True)
Out[23]: <AxesSubplot:>
```

Figure 3: Correlation between Y_pred and Y_test

References

[1] Numpy documentation. https://numpy.org/doc/stable/reference/,.

[2] Pandas documentation. https://pandas.pydata.org/docs/,.

[3] *Probabilistic Machine Learning An Introduction*. The MIT Press, 2022.

[4] ALGOR BRUCE. Kaggle dataset. https://www.kaggle.com/datasets/quantbruce/real-estate-price-prediction.

[5] Changyou Chen. 100 days of ml code. https://github.com/cchangyou/100-Days-Of-ML-Code/tree/master/Code.

[6] KD Nuggets Diego Lopez Yse. Kd nuggets your guide to linear regression models. https://www.kdnuggets.com/2020/10/guide-linear-regression-models.html,.

[7] Lauren Shin. Towards data science. https://towardsdatascience.com/graphs-and-linear-regression-734d1446e9cd.

Multiple Linear Regression Experiment Report

Darryl Vas Prabhu
Department of Computer Science and Engineering
University at Buffalo, Buffalo, NY 14260
dvasprab@buffalo.edu

1 Multiple Linear Regression

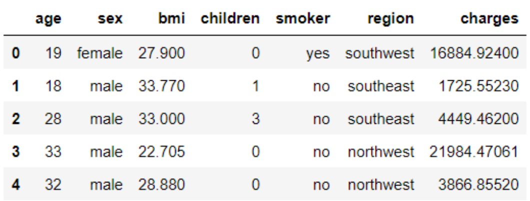
[1]Often times the prediction of a particular outcome depends not on one factor, but many factors. Say for example, the number of accidents in a city may be caused by bad road maintenance, bad vehicle maintenance, number of non operational traffic lights etc. All these factors can lead to an increase in the number of accidents. In short, the outcome of an incident(number of road accidents here) depends on various factors mentioned above. This leads us to a question of whether we can predict the number of accidents based on enough data-points from each factor. This is the concept of multiple linear regression.

In machine learning, the outcome is called the dependant variable and the factors affecting the outcome are called features or independent variables. In simple linear regression, we predict the output of the dependant variable $\mathbf{y}$ based on some function of a single independent variable $\mathbf{x}$. The same is true for multiple linear regression as well, but with many independent variables $\mathbf{x1}$, $\mathbf{x2}$, $\mathbf{x3}$, ...$\mathbf{xn}$. The equation of multiple linear regression is given by:

$$y = b0 + b1x1 + b2x2 + ... + bnxn$$

where y represents the dependent variable. $x, x1, x2, ..xn$ represents the independent variables. The intercept/bias, $b0$, represents the value of y when all features $x1, x2, x3, ..xn$ equals zero. $b1, b2, ..bn$ represents the regression coefficients of $x, x1, x2, ..xn$.

Since multiple regression model depends on many factors , visualizing the same in a 2-dimensional plane is not possible. However the machine is able to compute the model coefficients with training and test data provided. The goal of multiple regression is to find the coefficients such that the predicted value and actual output value are in close proximity. To understand multiple linear regression model, the **insurance** dataset from kaggle [5] was used to predict the medical charge based on many factors in the dataset. A snippet of the dataset is on the right and it's description is below:



	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

Figure 1: Multiple linear regression dataset snip

[1] Multiple Linear Regression.

2 Experiments

2.1 Decription of the dataset

The 6 items below are the input independant variables and the last one "charges" is the dependant output variable we are trying to predict.

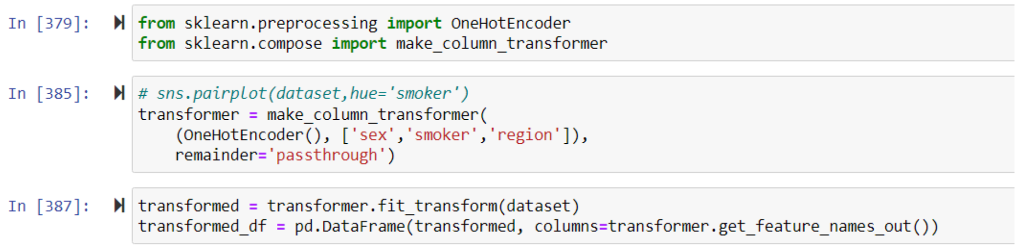
- age - Provides the age of various individuals.
- sex - Indicates the gender i.e : male or female.
- bmi - Provides body mass index of the individual.
- children - Provides the number of children the individual has.
- smoker - Indicates the individual smokes or not i.e : yes or no.
- region - Indicates area of region where he/she lives.
- charges - Provides the charges incurred.

2.2 Caveats

During the experiment, there were issues with categorical data being handled. Since categorical data needs to be converted or encoded, OneHotEncoder[4] was used on the dataset to change the independant categorical valued variables 'sex','smoker' and 'region' into numeric for analysis. This increased the total number of columns to 12 categories :

- onehotencoder__sex_female
- onehotencoder__sex_male
- onehotencoder__smoker_no
- onehotencoder__smoker_yes
- onehotencoder__region_northeast
- onehotencoder__region_northwest
- onehotencoder__region_southeast
- onehotencoder__region_southwest
- remainder__age
- remainder__bmi
- remainder__children
- remainder__charges

The ones beginning with "onehotencoder" are the encoded variables. One of the columns was dropped to prevent the dummy variable trap while selecting the trainging dataset.



```
In [379]:  from sklearn.preprocessing import OneHotEncoder
            from sklearn.compose import make_column_transformer

In [385]:  # sns.pairplot(dataset,hue='smoker')
            transformer = make_column_transformer(
                (OneHotEncoder(), ['sex','smoker','region']),
                remainder='passthrough')

In [387]:  transformed = transformer.fit_transform(dataset)
            transformed_df = pd.DataFrame(transformed, columns=transformer.get_feature_names_out())
```

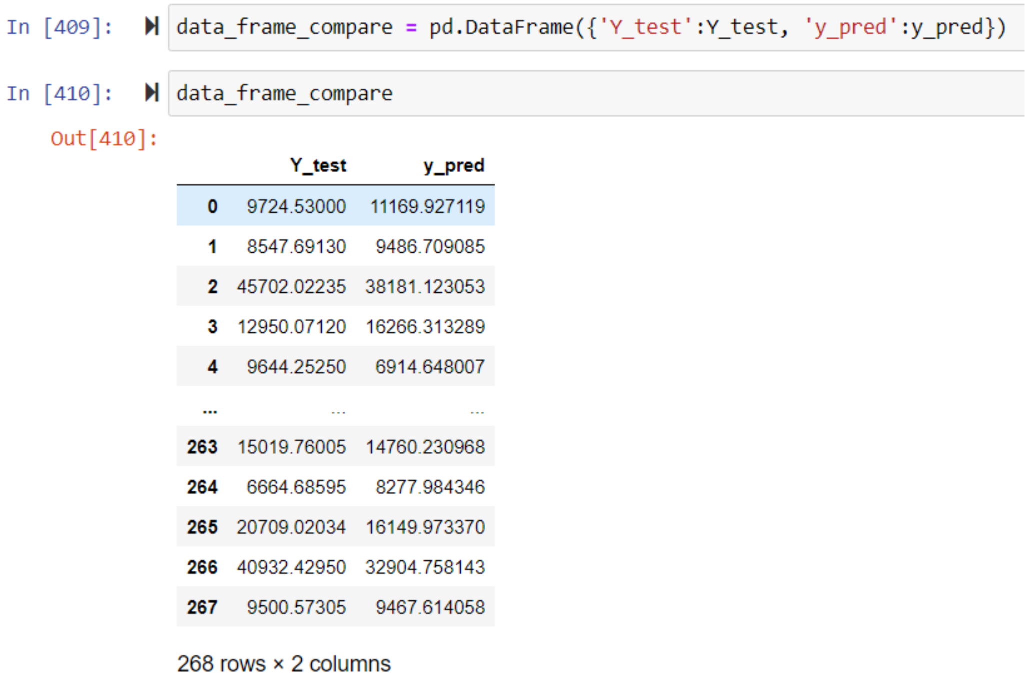
Figure 2: One Hot Encoder Transformation

2.3 Results

After encoding the data, the data was split into 4 parts. X_train , Y_train for training the datasets. X_test, Y_test for testing the dataset.

It is a general rule of thumb to have 80% of the data undergo training, and test the trained regressor model on the then remaining 20% test data which the machine has not seen before. The model was trained using scikit's machine learning package class LinearRegression()

The result of the test proved that the prediction values of charges are not too far from the test data charges.



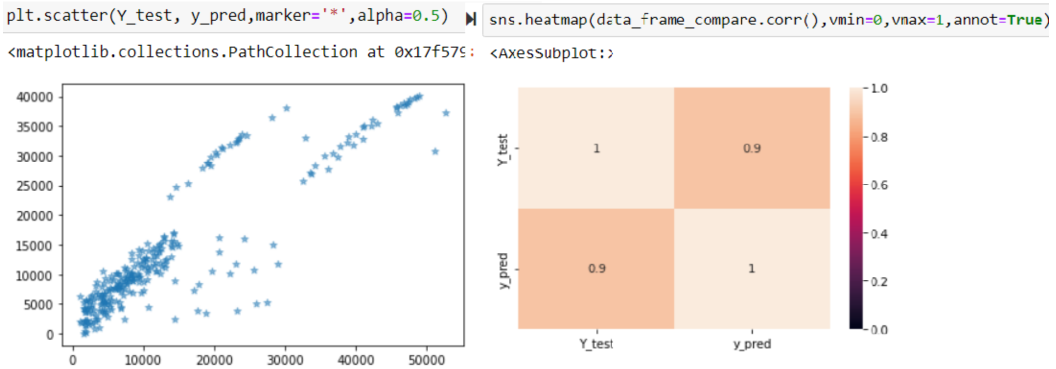
```
In [409]:  data_frame_compare = pd.DataFrame({'Y_test':Y_test, 'y_pred':y_pred})

In [410]:  data_frame_compare
Out[410]:
```

	Y_test	y_pred
0	9724.53000	11169.927119
1	8547.69130	9486.709085
2	45702.02235	38181.123053
3	12950.07120	16266.313289
4	9644.25250	6914.648007
...	...	...
263	15019.76005	14760.230968
264	6664.68595	8277.984346
265	20709.02034	16149.973370
266	40932.42950	32904.758143
267	9500.57305	9467.614058

268 rows × 2 columns

Figure 3: Y test vs Y pred



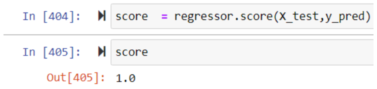
```
plt.scatter(Y_test, y_pred,marker='*',alpha=0.5)
<matplotlib.collections.PathCollection at 0x17f579

sns.heatmap(data_frame_compare.corr(),vmin=0,vmax=1,annot=True)
<AxesSubplot:>
```

Figure 4: Correlation between test result and predicted result

The regression score which is the score used to measure how the model predicts the data which in our case is 1.

In addition we can see the heatmap and the scatter plot which show the high correlation between the predicted value of **charge** and the actual test value of **charges**.



```
In [404]:   score  = regressor.score(X_test,y_pred)

In [405]:   score
  Out[405]: 1.0
```

Figure 5: Regression score

References

[1] Numpy documentation. `https://numpy.org/doc/stable/reference/`.

[2] Pandas documentation. `https://pandas.pydata.org/docs/`.

[3] *Probabilistic Machine Learning An Introduction*. The MIT Press, 2022.

[4] The DataGY. One hot encoding. `https://datagy.io/sklearn-one-hot-encode/`.

[5] Kaggle Datasets. Medical cost personal datasets. `https://www.kaggle.com/datasets/mirichoi0218/insurance`.

[6] Scikit-Learn. Scikit learn documentation. `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html`.

Logistic Regression Experiment Report

Darryl Vas Prabhu
Department of Computer Science and Engineering
University at Buffalo, Buffalo, NY 14260
dvasprab@buffalo.edu

1 Logistic Regression

[1]There are certain problems in machine learning where one needs to perform a classification based on the input variables. For example, if we have images with a certain feature, then classifying those images in certain classes is a problem logistic regression aims to solve.

In machine learning, the outcome is a class variable. Ex: It can be a yes or no. Or it can be a categorized label with labels from 1 to C. Hence the logistic regression model is called a discrimative classification model. It is most often used in machine learning to predict a where a group of inputs fit.

If there are only 2 classes, i.e. C = 2, then it is known as binary logistic regression. The formula for binary classification regression output is given as follows :

$$z = w^T x + b$$

where y is given by the sigmoid function

$$y = 1/(1 + e^{-z})$$

y represents the dependent variable. x represents the independent variables. The intercept/bias, b, represents the value of y when all features $x1, x2, x3, ..xn$ equals zero. W^T represents the weights of the linear classifier.In general we are trying to fetch the probability of the class when there is a success rate i.e. P[y=1] Mot often the class is defined using the sigmoid function and the threshold for setting the class is usually taken at probability of P(y=0.5) where the graph changes it's direction as denoted in the sigmoid function.

The goal of logistic regression is to classify outputs based on given input variables. To understand Logistic Regression model, the **insurance** dataset from kaggle [4] was used to predict the loan status for individuals based on their income and co-applicants income. A snippet of the dataset and it's description is below:

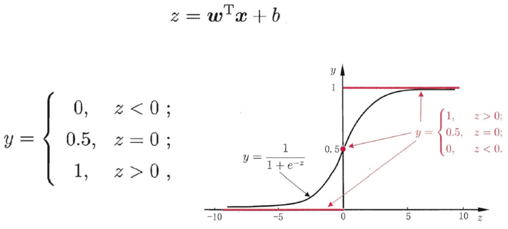
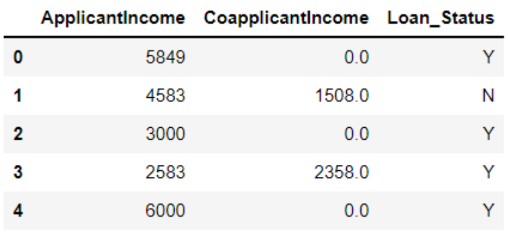


Figure 1: Logistic Regression model

[1] Logistic Regression.



	ApplicantIncome	CoapplicantIncome	Loan_Status
0	5849	0.0	Y
1	4583	1508.0	N
2	3000	0.0	Y
3	2583	2358.0	Y
4	6000	0.0	Y

Figure 2: Dataset Snip

2 Experiments

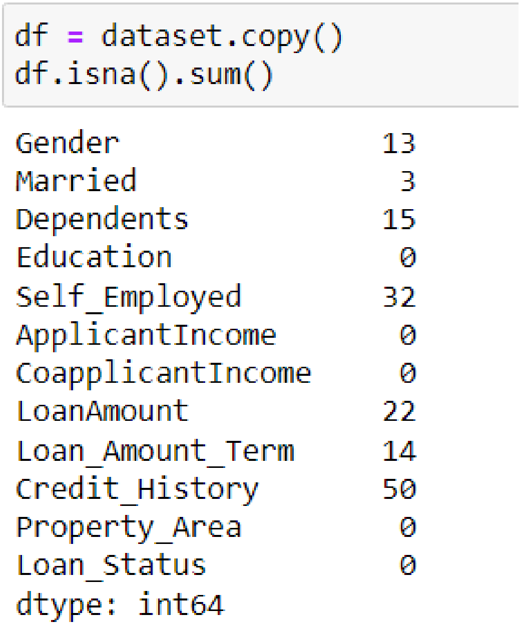
2.1 Decription of the dataset

The 2 items below are the input independant variables and the last one "Loan_Status" is the dependant output variable we are trying to predict.

- ApplicantIncome - Provides the Applicant's income.
- CoapplicantIncome - Provides the Coapplicant's income
- Loan_Status - Provides the label where the loan is sanctioned or not.

2.2 Caveats

Many data items were having missing values or nan values in the data due to which these items were ruled out for considering as variables for analysis. In order to find the true dependance on the income factor only the income variables were taken into consideration for the input variables.



```
df = dataset.copy()
df.isna().sum()
```

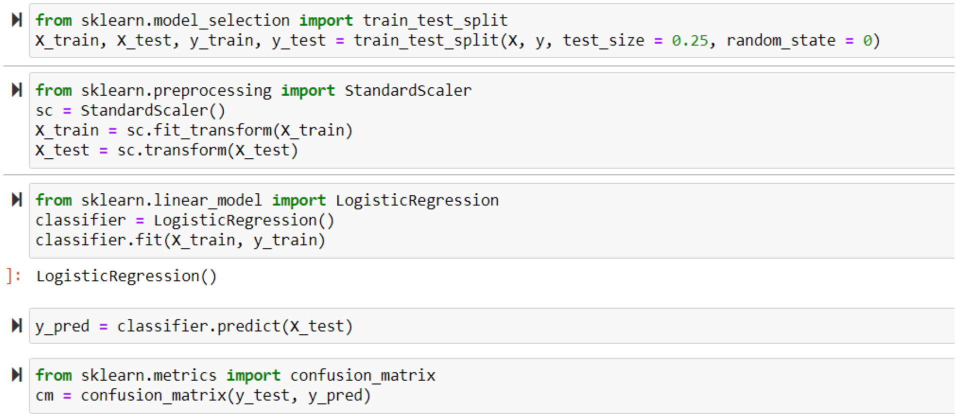
```
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

Figure 3: Data having null values

2.3 Results

The data was passed to split using scikit-learns train_test_split method from the model_selection package. ApplicantIncome and CoapplicantIncome were used for determining the Loan_Status label here.

Since the range of values in the independant variables can vary, in order to get it to the normal form so that the values are standardized, feature scaling was used as below using the StandardScaler() method. This will bring the values in the same range so that one feature does not overestimate the other and the model is trained appropriately. If a feature has a variance that is orders of magnitude larger than others, it might dominate the objective function and make the estimator unable to learn from other features correctly as expected.



```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
classifier.fit(X_train, y_train)
```

```
]: LogisticRegression()
```

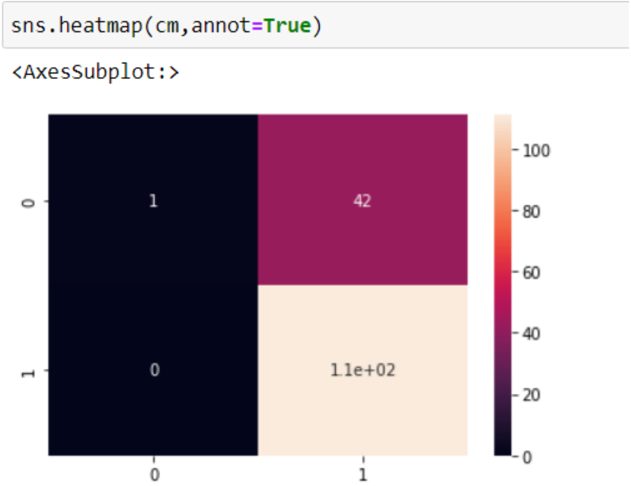
```
y_pred = classifier.predict(X_test)
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

Figure 4: Standard Scaler and Logistic Regression class

The result of the Logistic Regression proved that the prediction of loan status are somewhat varied with 112 out of 154 predicted same as the test values and the remaining 42 are not predicted appropriately.

The confusion matrix gives us a total True score of 112 which is approximately 72% correct. The heatmap indicates the same as belwo:



```
sns.heatmap(cm,annot=True)
```

```
<AxesSubplot:>
```

Figure 5: Confusion Matrix

References

[1] Numpy documentation. https://numpy.org/doc/stable/reference/.

[2] Pandas documentation. https://pandas.pydata.org/docs/.

[3] *Probabilistic Machine Learning An Introduction*. The MIT Press, 2022.

[4] Kaggle Datasets. https://www.kaggle.com/code/drfrank/loan-data-visualisation-eda-machine-learning/data.

[5] Sklearn documentation. Standardscaler. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html.

[6] Scikit-Learn. Scikit learn documentation. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html.