Kmit

SONET

Data Science

(Level-I)

Ajeet K. Jain, M. Narsimlu

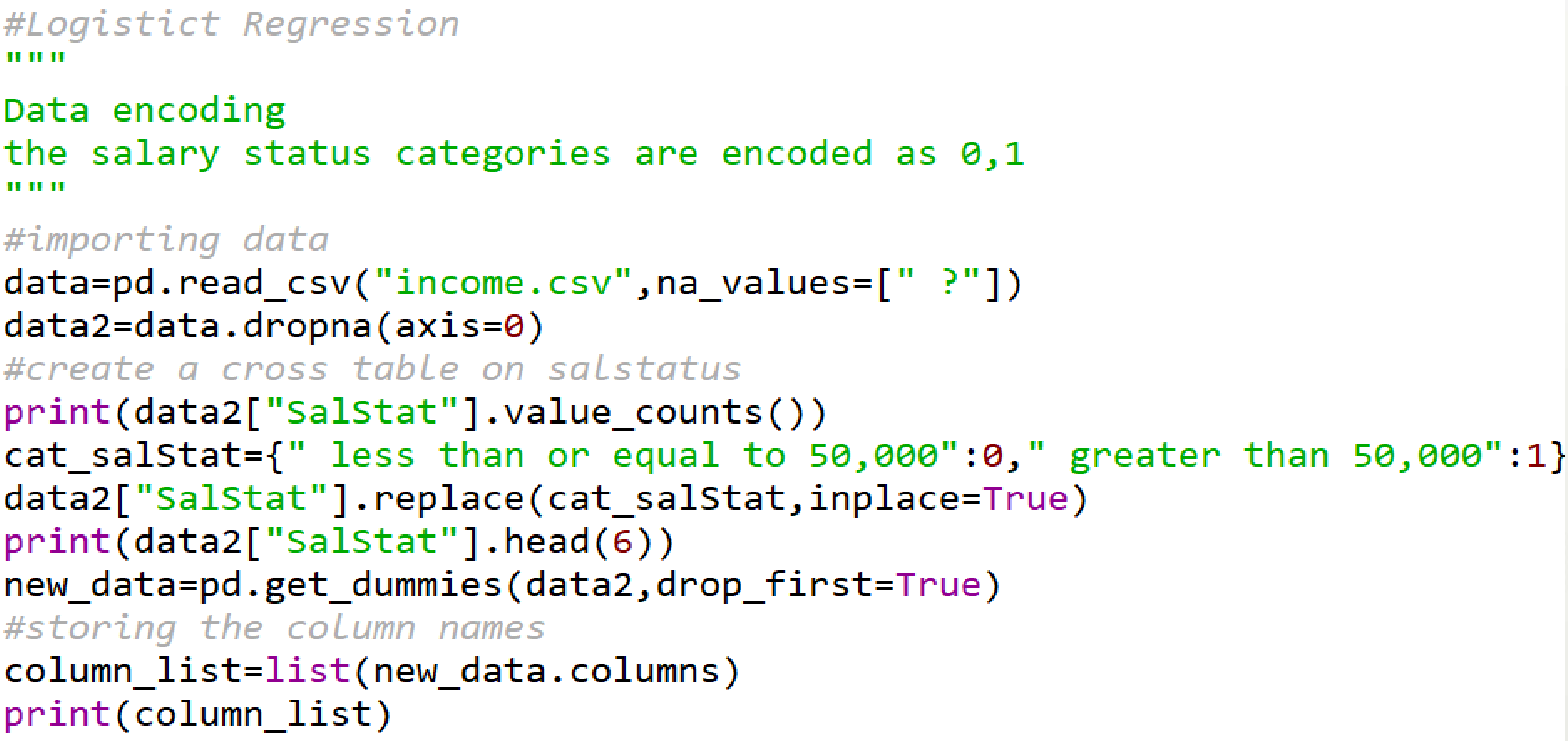
(ML TEAM)- SONET, KMIT, Hyderabad

Session – 32

This session deals with

Introduction to Case Study

Data Encoding



```
#Logistict Regression
"""
Data encoding
the salary status categories are encoded as 0,1
"""
#importing data
data=pd.read_csv("income.csv",na_values=[" ?"])
data2=data.dropna(axis=0)
#create a cross table on salstatus
print(data2["SalStat"].value_counts())
cat_salStat={" less than or equal to 50,000":0," greater than 50,000":1}
data2["SalStat"].replace(cat_salStat,inplace=True)
print(data2["SalStat"].head(6))
new_data=pd.get_dummies(data2,drop_first=True)
#storing the column names
column_list=list(new_data.columns)
print(column_list)
```

Classification

We use the training dataset to get better boundary conditions which could be used to determine each target class.

Once the boundary conditions are determined, the next task is to predict the target class. This process is known as classification.

Analysis of the income data to predict whether the individual person is miss using salary **(Target class: Yes or No)**

Classifying salary status from features like age, occupation, educatio etc.. **(Target classes: less than or equal to 50k,greater than 50k)**

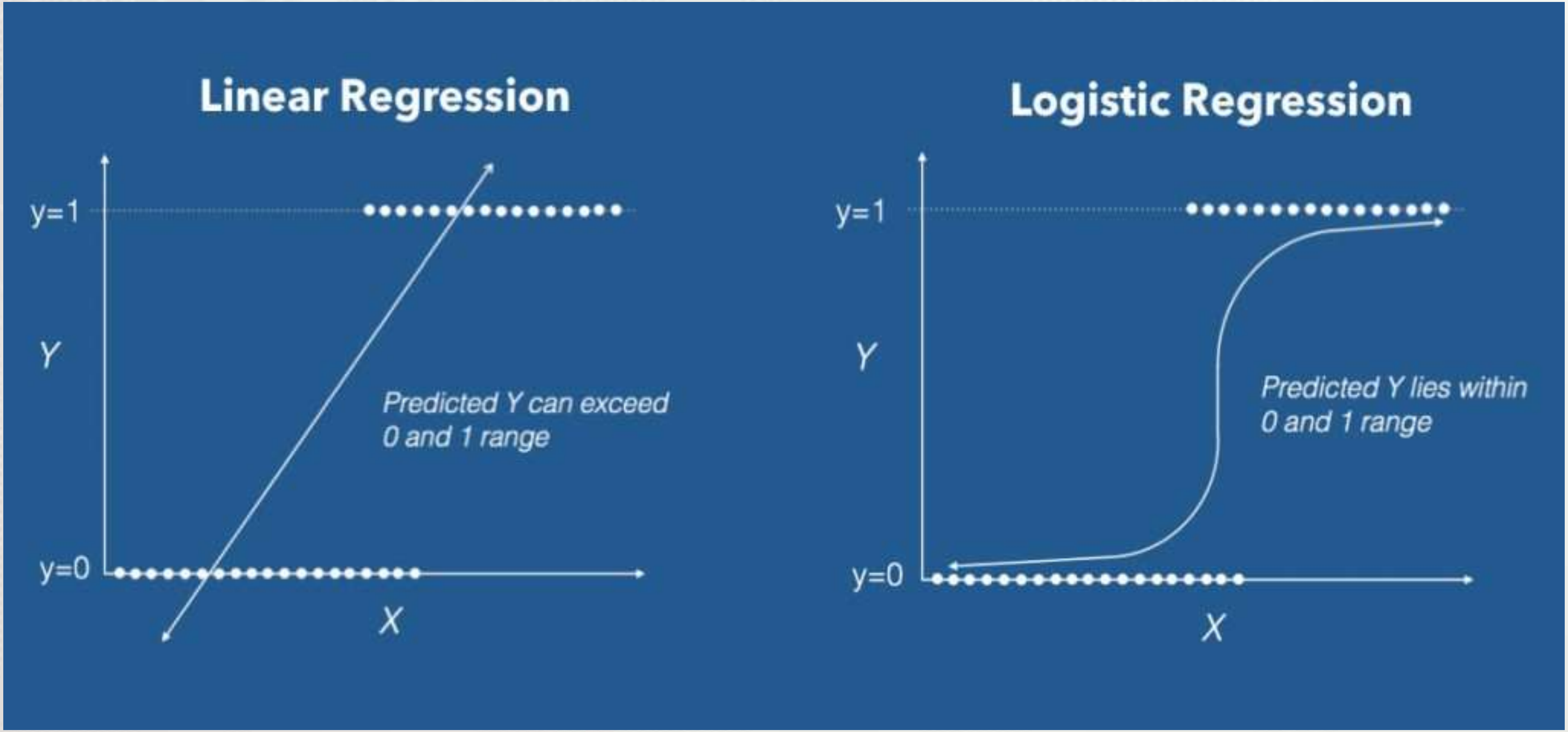
Why not linear regression?

When the response variable has only 2 possible values, it is desirable to have a model that predicts the value either as 0 or 1 or as a probability score that ranges between 0 and 1.

Linear regression does *not* have this capability.

If you use linear regression to model a binary response variable, the resulting model may not restrict the predicted Y values within 0 and 1.

Why not linear regression?


Linear Regression
y=1
Y
y=0
X
Predicted Y can exceed 0 and 1 range
Logistic Regression
y=1
Y
y=0
X
Predicted Y lies within 0 and 1 range

Logistic Regression

- Logistic regression is a classification algorithm used to assign observations to a discrete set of classes.
- Examples:
- **Spam Detection :** Predicting if an email is Spam or not
- **Credit Card Fraud :** Predicting if a given credit card transaction is fraud or not
- **Health :** Predicting if a given mass of tissue is benign or malignant
- **Marketing :** Predicting if a given user will buy an insurance product or not
- **Banking :** Predicting if a customer will default on a loan.
- Logistic regression transforms its output using the logistic sigmoid function to return a probability value.

Types Logistic Regressions

1.Binary (eg. Less than or equal to 50k or greater than 50k)
2.Multi-linear functions Class (eg. Short loan,middle term loan,long term loan)

it is a predictive analysis algorithm and based on the concept of probability.

Logistic Regression

We can call a Logistic Regression a Linear Regression model but the Logistic Regression uses a more complex cost function

The cost function is‘**Sigmoid function**’ or also known as the ‘logistic function’ instead of a linear function.

The hypothesis of logistic regression tends it to limit the cost function between 0 and 1.

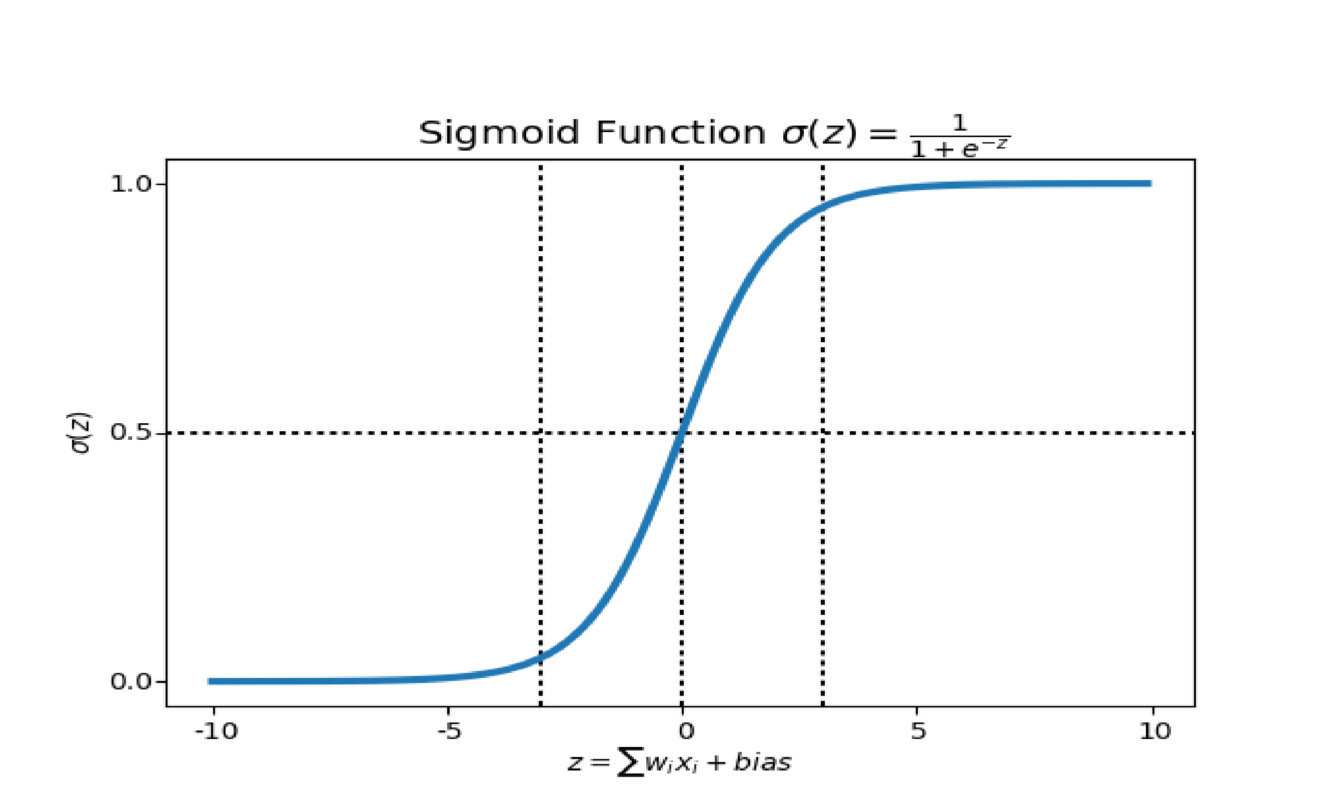
$$0 \leq h_\theta(x) \leq 1$$

Sigmoid Function

The function maps any real value into another value between 0 and 1.

In machine learning, we use sigmoid to map predictions to probabilities.

The hypothesis of logistic regression tends it to limit the cost function between 0 and 1.



Sigmoid Function

$$f(x) = \frac{1}{1 + \mathrm{e}^{-(x)}}$$

Sigmoid Function

linear regression we used a formula of the hypothesis i.e..

$$h\Theta(x) = \beta_0 + \beta_1 X$$

logistic regression.

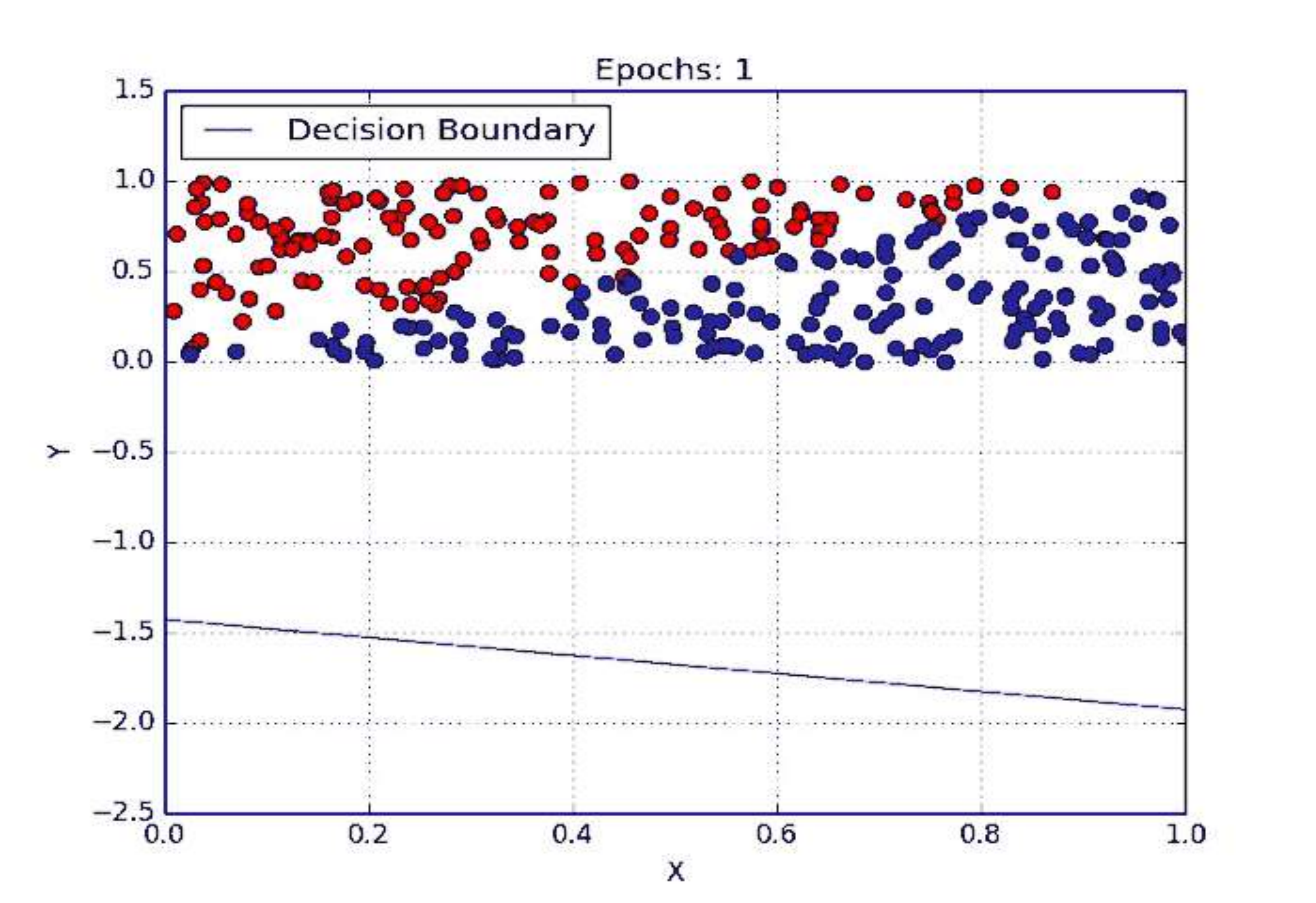
$$\sigma(Z) = \sigma(\beta_0 + \beta_1 X)$$

expected that our hypothesis will give values between 0 and 1

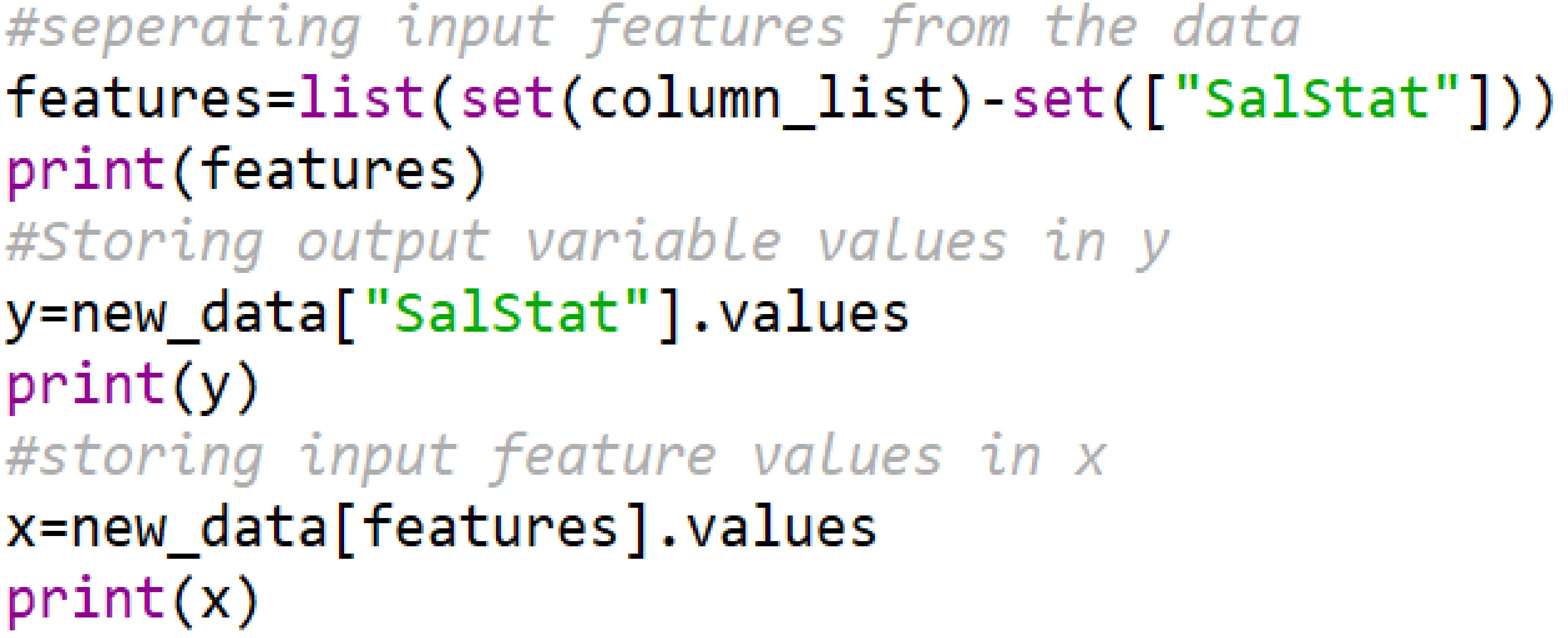
$Z = \beta_0 + \beta_1 X$
$h\Theta(x) = \text{sigmoid}(Z)$
i.e. $h\Theta(x) = 1/(1 + e\text{^}-(\beta_0 + \beta_1 X)$

$$h\theta(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

Logistic Regression

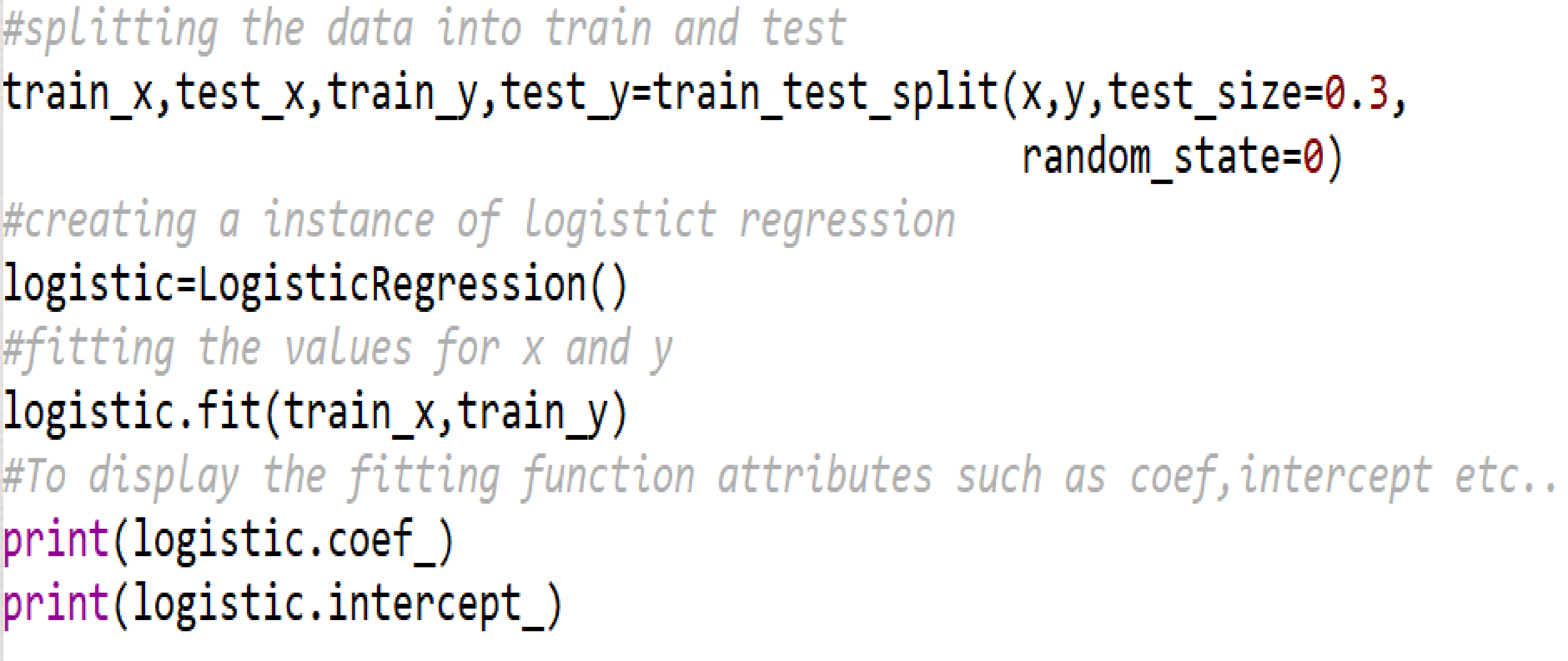


Identifying input features



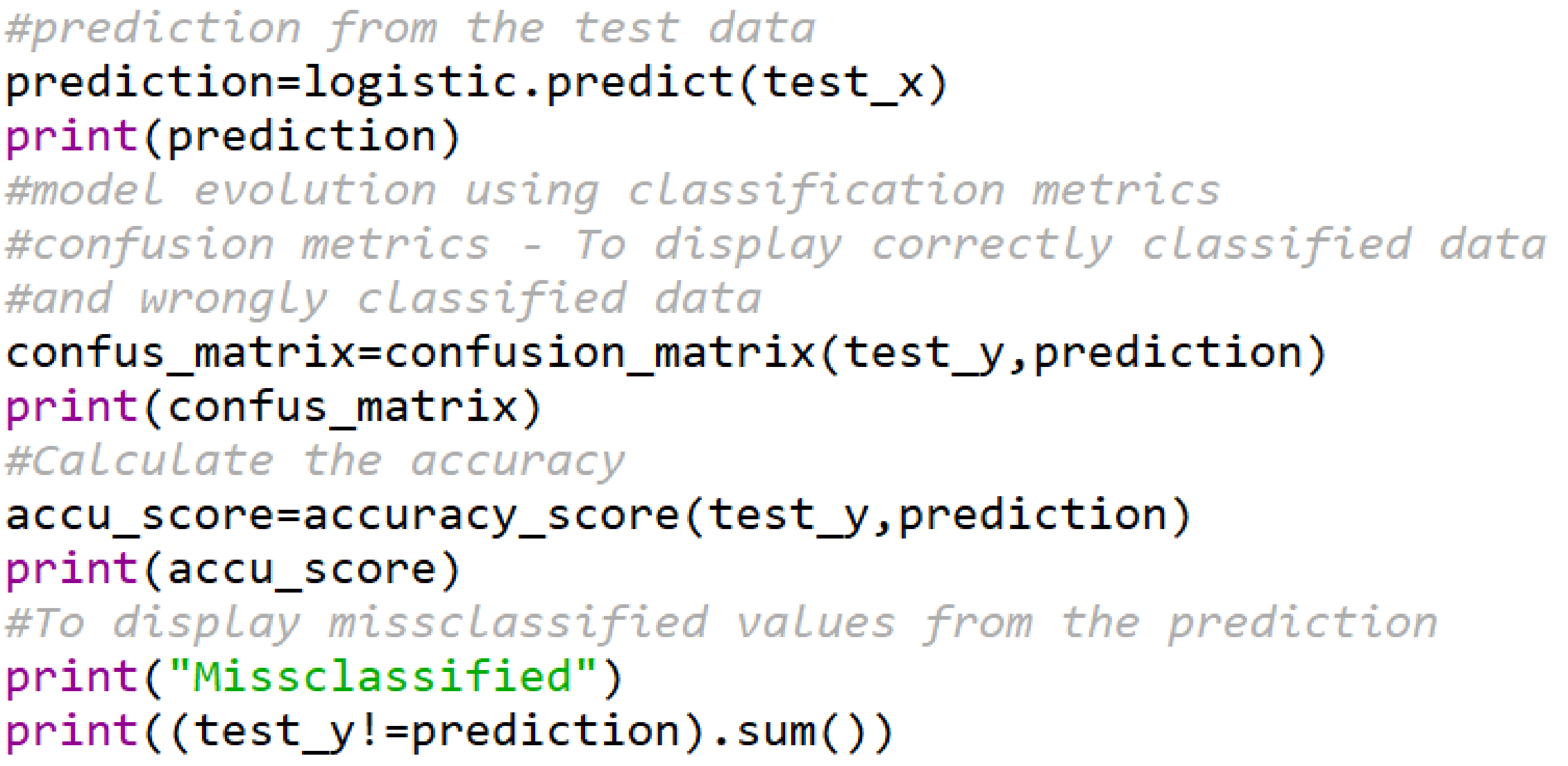
```
#seperating input features from the data
features=list(set(column_list)-set(["SalStat"]))
print(features)
#Storing output variable values in y
y=new_data["SalStat"].values
print(y)
#storing input feature values in x
x=new_data[features].values
print(x)
```

Splitting Data and Model fitting



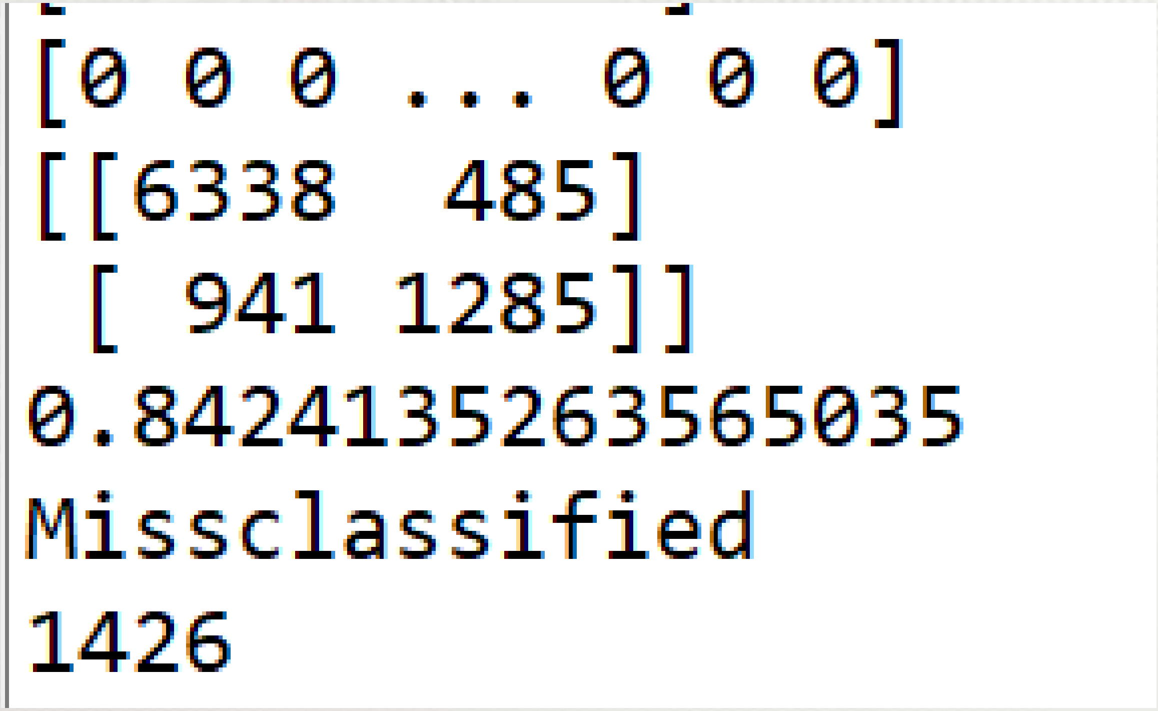
```
#splitting the data into train and test
train_x,test_x,train_y,test_y=train_test_split(x,y,test_size=0.3,
                                                random_state=0)
#creating a instance of logistict regression
logistic=LogisticRegression()
#fitting the values for x and y
logistic.fit(train_x,train_y)
#To display the fitting function attributes such as coef,intercept etc..
print(logistic.coef_)
print(logistic.intercept_)
```

Evaluating the model



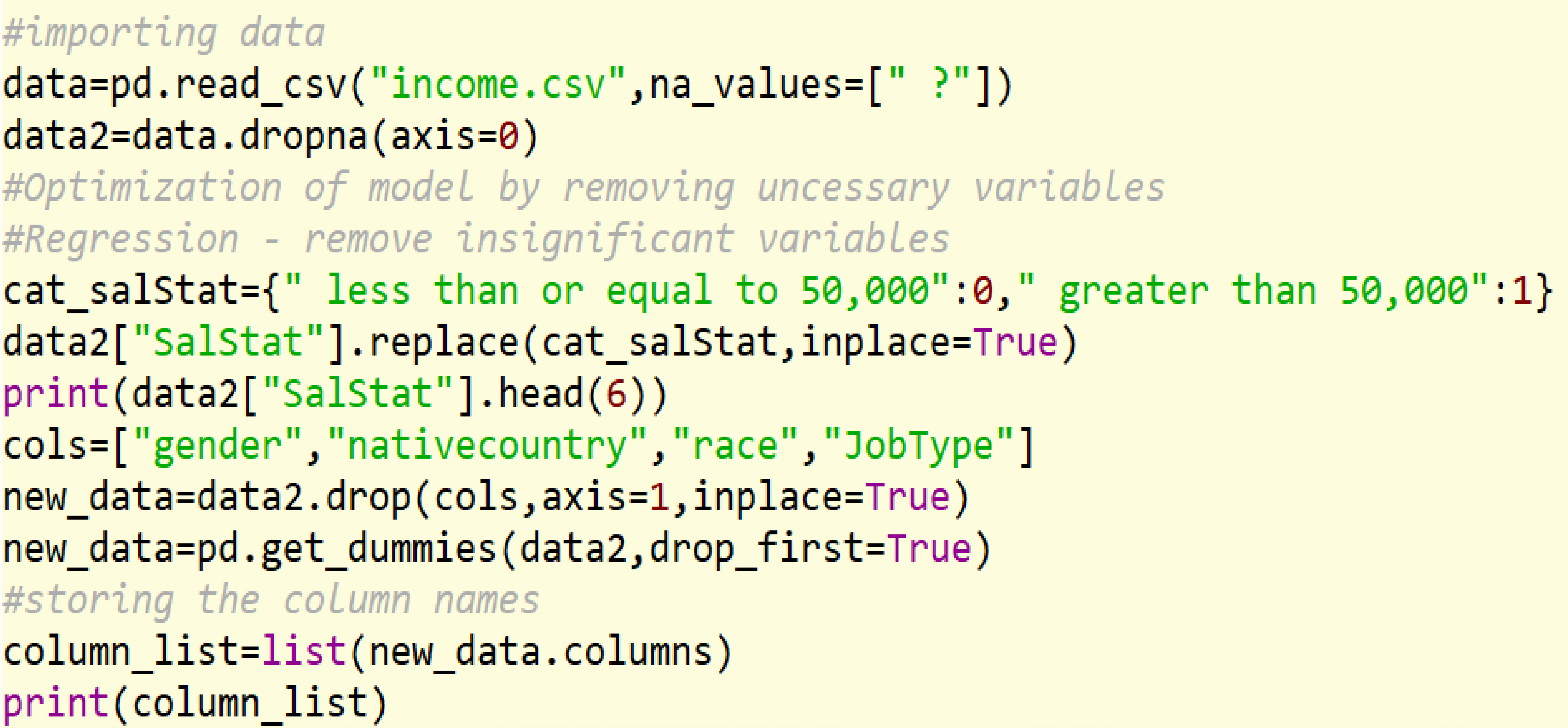
```
#prediction from the test data
prediction=logistic.predict(test_x)
print(prediction)
#model evolution using classification metrics
#confusion metrics - To display correctly classified data
#and wrongly classified data
confus_matrix=confusion_matrix(test_y,prediction)
print(confus_matrix)
#Calculate the accuracy
accu_score=accuracy_score(test_y,prediction)
print(accu_score)
#To display missclassified values from the prediction
print("Missclassified")
print((test_y!=prediction).sum())
```

Output



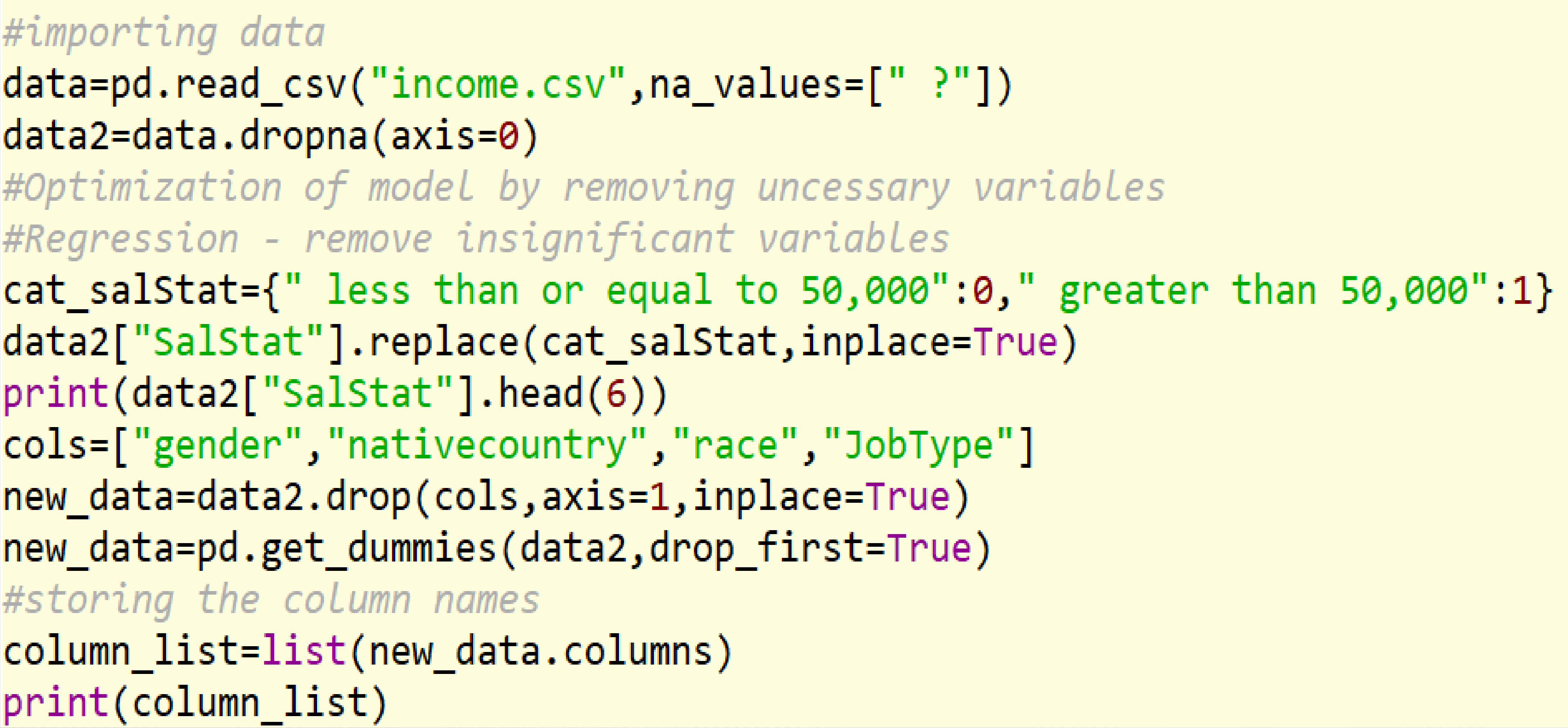
```
[0 0 0 ... 0 0 0]
[[6338  485]
 [ 941 1285]]
0.8424135263565035
Missclassified
1426
```

Optimization of model



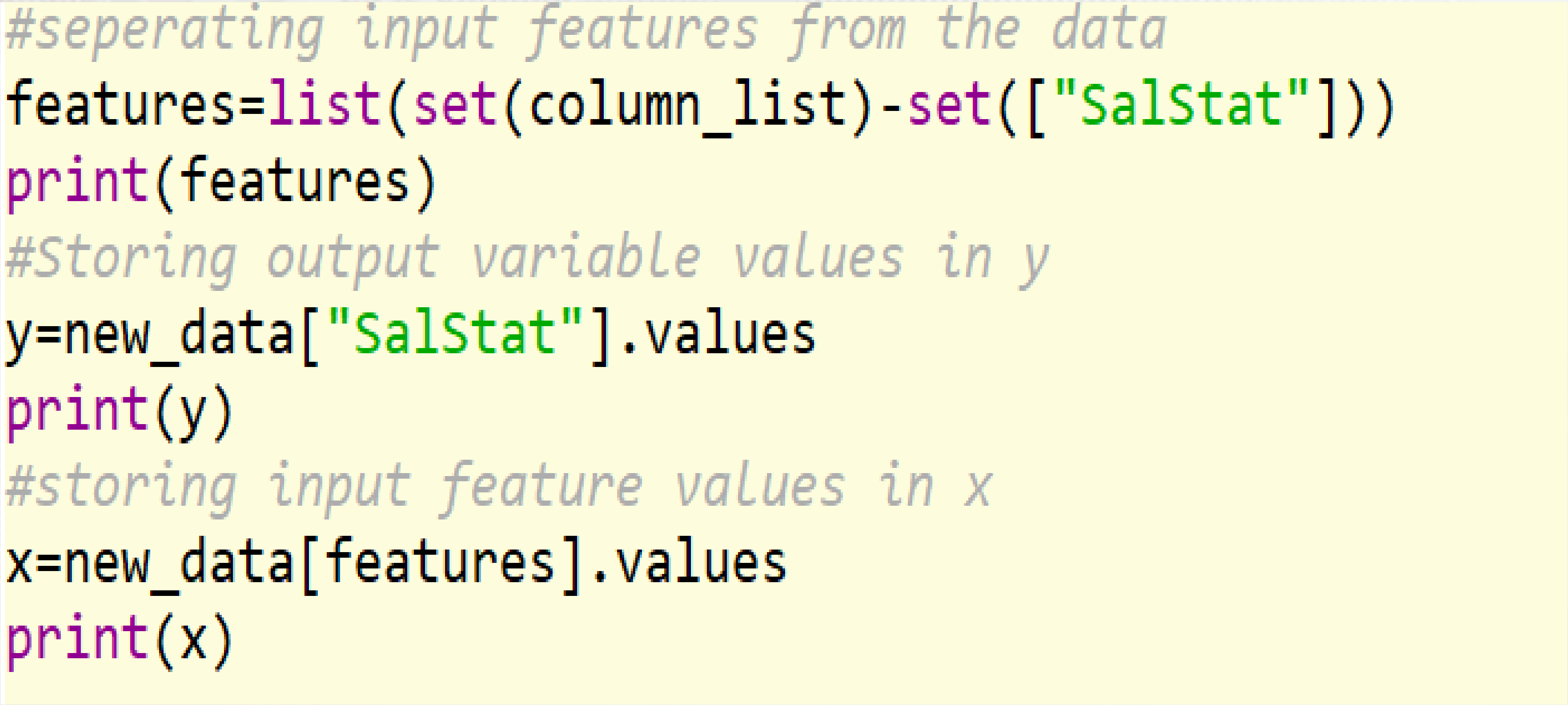
```
#importing data
data=pd.read_csv("income.csv",na_values=[" ?"])
data2=data.dropna(axis=0)
#Optimization of model by removing uncessary variables
#Regression - remove insignificant variables
cat_salStat={" less than or equal to 50,000":0," greater than 50,000":1}
data2["SalStat"].replace(cat_salStat,inplace=True)
print(data2["SalStat"].head(6))
cols=["gender","nativecountry","race","JobType"]
new_data=data2.drop(cols,axis=1,inplace=True)
new_data=pd.get_dummies(data2,drop_first=True)
#storing the column names
column_list=list(new_data.columns)
print(column_list)
```

Optimization of model



```
#importing data
data=pd.read_csv("income.csv",na_values=[" ?"])
data2=data.dropna(axis=0)
#Optimization of model by removing uncessary variables
#Regression - remove insignificant variables
cat_salStat={" less than or equal to 50,000":0," greater than 50,000":1}
data2["SalStat"].replace(cat_salStat,inplace=True)
print(data2["SalStat"].head(6))
cols=["gender","nativecountry","race","JobType"]
new_data=data2.drop(cols,axis=1,inplace=True)
new_data=pd.get_dummies(data2,drop_first=True)
#storing the column names
column_list=list(new_data.columns)
print(column_list)
```

Identifying the features

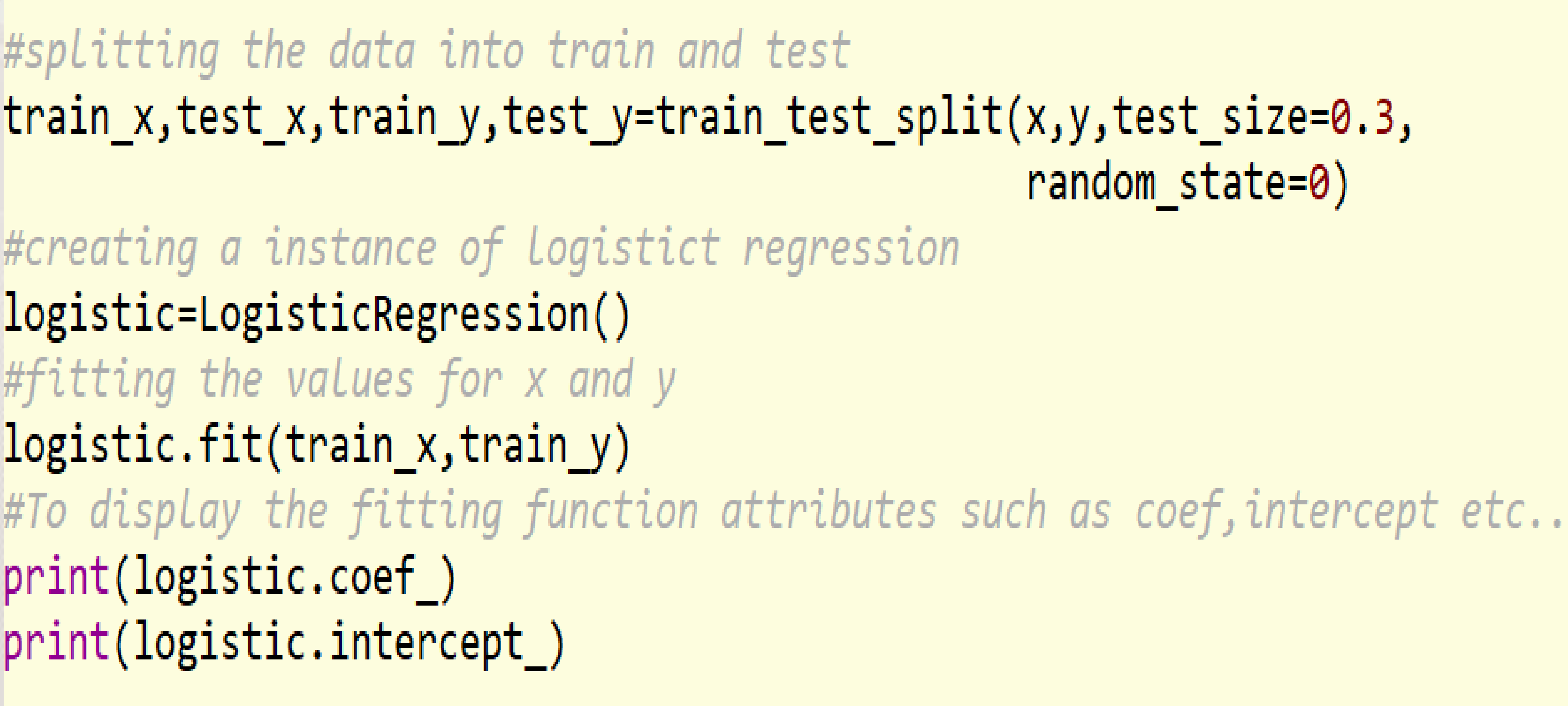


```
#seperating input features from the data
features=list(set(column_list)-set(["SalStat"]))
print(features)
#Storing output variable values in y
y=new_data["SalStat"].values
print(y)
#storing input feature values in x
x=new_data[features].values
print(x)
```

Kmit SONET

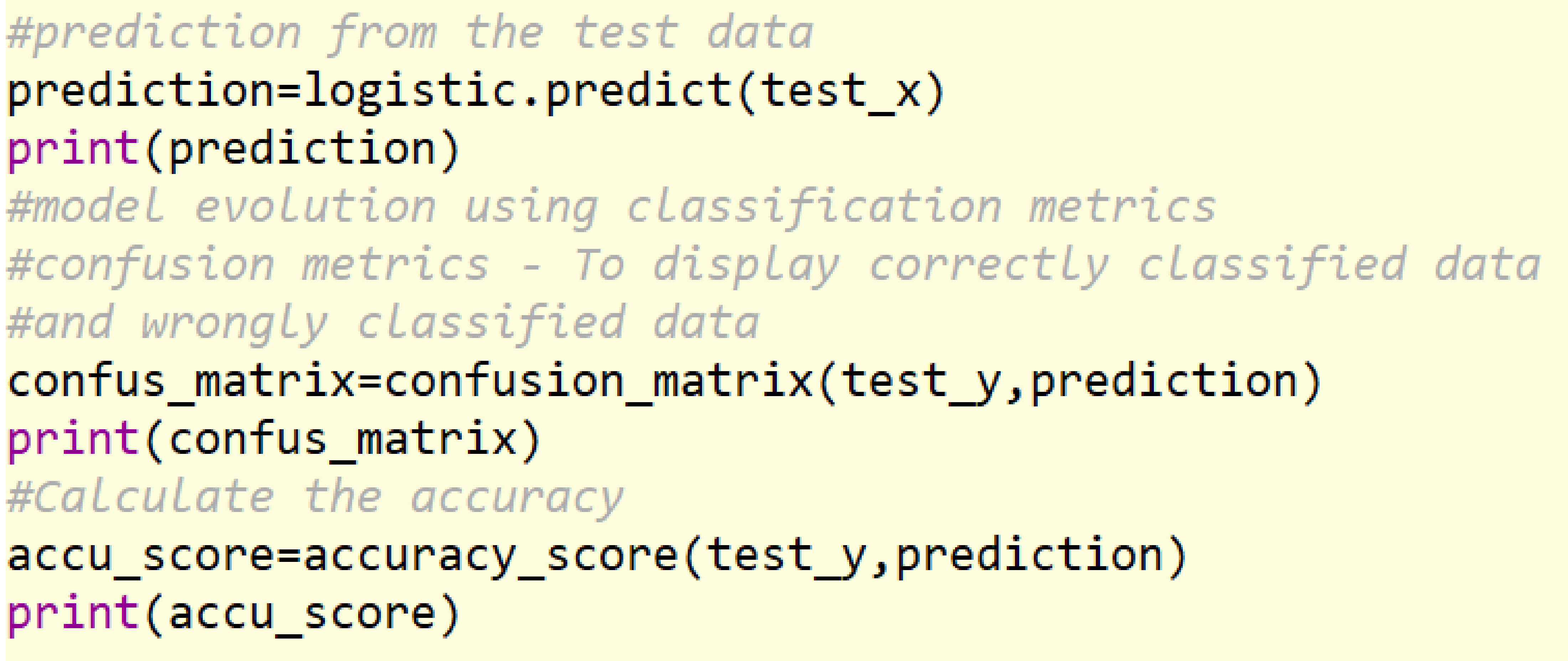
Model fitting

DATA SCIENCE



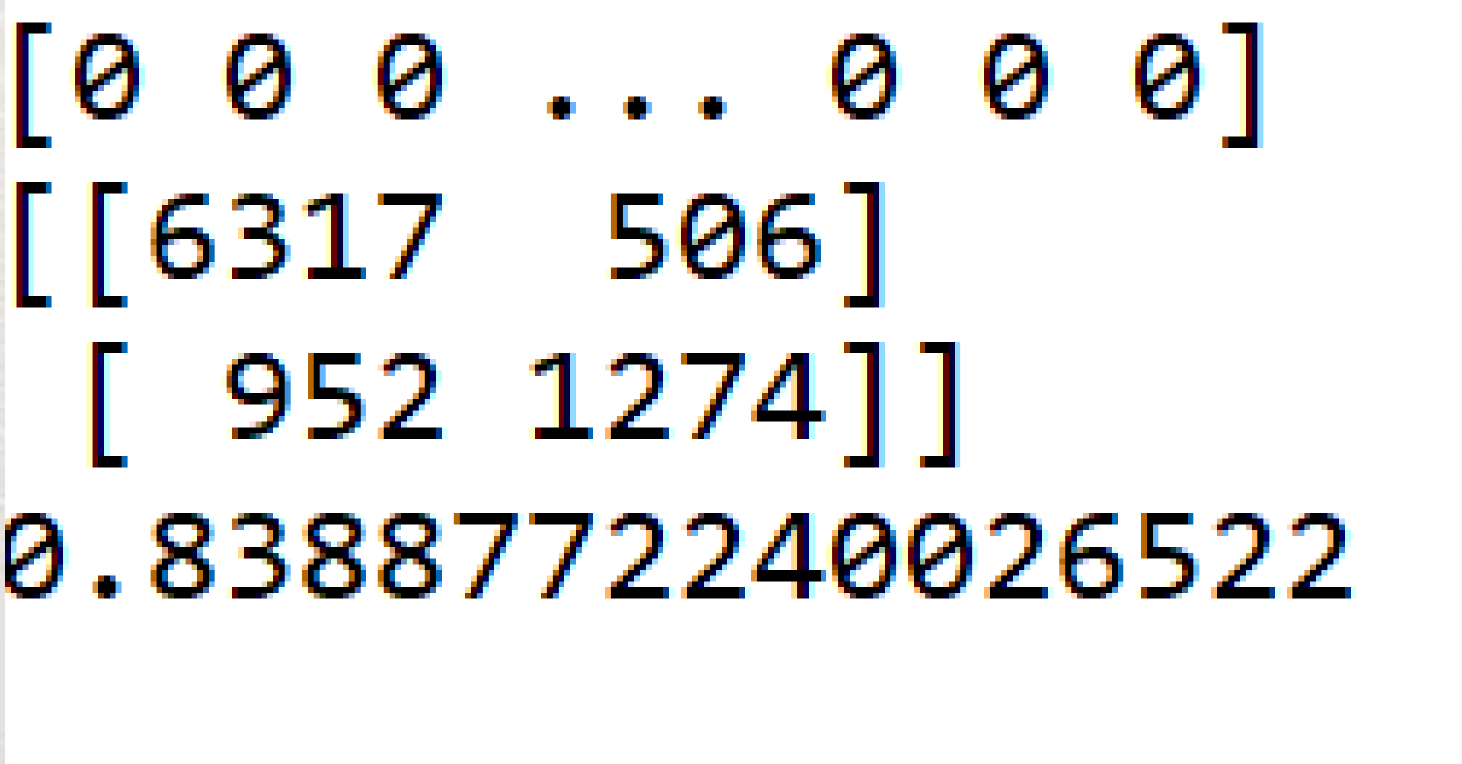
```
#splitting the data into train and test
train_x,test_x,train_y,test_y=train_test_split(x,y,test_size=0.3,
                                               random_state=0)
#creating a instance of logistict regression
logistic=LogisticRegression()
#fitting the values for x and y
logistic.fit(train_x,train_y)
#To display the fitting function attributes such as coef,intercept etc..
print(logistic.coef_)
print(logistic.intercept_)
```

Evaluating the model



```
#prediction from the test data
prediction=logistic.predict(test_x)
print(prediction)
#model evolution using classification metrics
#confusion metrics - To display correctly classified data
#and wrongly classified data
confus_matrix=confusion_matrix(test_y,prediction)
print(confus_matrix)
#Calculate the accuracy
accu_score=accuracy_score(test_y,prediction)
print(accu_score)
```

Output



```
[0 0 0 ... 0 0 0]
[[6317  506]
 [ 952 1274]]
0.8388772240026522
```

Conclusion

You are aware of

Data Encoding

Classification case study

We will proceed with

Case Study

THANK YOU