## Log Regression

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as ply  
import seaborn as sns

df = pd.read\_csv('cust\_satisfaction.csv')  
df.head()

Gender Customer Type Type of Travel Class \  
0 Male Loyal Customer Personal Travel Eco Plus   
1 Male disloyal Customer Business travel Business   
2 Female Loyal Customer Business travel Business   
3 Female Loyal Customer Business travel Business   
4 Male Loyal Customer Business travel Business   
  
 satisfaction Age Flight Distance Inflight entertainment \  
0 neutral or dissatisfied 13 460 5   
1 neutral or dissatisfied 25 235 1   
2 satisfied 26 1142 5   
3 neutral or dissatisfied 25 562 2   
4 satisfied 61 214 3   
  
 Baggage handling Cleanliness Departure Delay in Minutes \  
0 4 5 25   
1 3 1 1   
2 4 5 0   
3 3 2 11   
4 4 3 0   
  
 Arrival Delay in Minutes   
0 18.0   
1 6.0   
2 0.0   
3 9.0   
4 0.0

#y --> customer types  
# balance and inbalance dataset  
df['Customer Type'].value\_counts()

Customer Type  
Loyal Customer 84923  
disloyal Customer 18981  
Name: count, dtype: int64

df.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 103904 entries, 0 to 103903  
Data columns (total 12 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 Gender 103904 non-null object   
 1 Customer Type 103904 non-null object   
 2 Type of Travel 103904 non-null object   
 3 Class 103904 non-null object   
 4 satisfaction 103904 non-null object   
 5 Age 103904 non-null int64   
 6 Flight Distance 103904 non-null int64   
 7 Inflight entertainment 103904 non-null int64   
 8 Baggage handling 103904 non-null int64   
 9 Cleanliness 103904 non-null int64   
 10 Departure Delay in Minutes 103904 non-null int64   
 11 Arrival Delay in Minutes 103594 non-null float64  
dtypes: float64(1), int64(6), object(5)  
memory usage: 9.5+ MB

df.isnull().sum()

Gender 0  
Customer Type 0  
Type of Travel 0  
Class 0  
satisfaction 0  
Age 0  
Flight Distance 0  
Inflight entertainment 0  
Baggage handling 0  
Cleanliness 0  
Departure Delay in Minutes 0  
Arrival Delay in Minutes 310  
dtype: int64

df[df['Arrival Delay in Minutes'].isnull()]

Gender Customer Type Type of Travel Class \  
213 Female Loyal Customer Business travel Eco   
1124 Male Loyal Customer Personal Travel Eco   
1529 Male Loyal Customer Business travel Business   
2004 Female disloyal Customer Business travel Business   
2108 Female Loyal Customer Personal Travel Eco   
... ... ... ... ...   
102067 Male Loyal Customer Personal Travel Eco Plus   
102384 Male Loyal Customer Business travel Eco   
102552 Female disloyal Customer Business travel Eco   
102960 Male Loyal Customer Business travel Eco   
103540 Female Loyal Customer Personal Travel Eco   
  
 satisfaction Age Flight Distance Inflight entertainment \  
213 satisfied 38 109 5   
1124 neutral or dissatisfied 53 1012 4   
1529 neutral or dissatisfied 39 733 2   
2004 neutral or dissatisfied 26 1035 2   
2108 neutral or dissatisfied 24 417 5   
... ... ... ... ...   
102067 neutral or dissatisfied 49 1249 3   
102384 neutral or dissatisfied 58 733 3   
102552 neutral or dissatisfied 29 1107 5   
102960 satisfied 58 1088 5   
103540 neutral or dissatisfied 33 359 4   
  
 Baggage handling Cleanliness Departure Delay in Minutes \  
213 4 5 31   
1124 4 4 38   
1529 2 3 11   
2004 4 2 41   
2108 2 5 1   
... ... ... ...   
102067 4 3 230   
102384 2 3 55   
102552 5 5 0   
102960 5 5 0   
103540 5 4 42   
  
 Arrival Delay in Minutes   
213 NaN   
1124 NaN   
1529 NaN   
2004 NaN   
2108 NaN   
... ...   
102067 NaN   
102384 NaN   
102552 NaN   
102960 NaN   
103540 NaN   
  
[310 rows x 12 columns]

df.dropna(inplace=True)

df.isnull().sum()

Gender 0  
Customer Type 0  
Type of Travel 0  
Class 0  
satisfaction 0  
Age 0  
Flight Distance 0  
Inflight entertainment 0  
Baggage handling 0  
Cleanliness 0  
Departure Delay in Minutes 0  
Arrival Delay in Minutes 0  
dtype: int64

df.duplicated().sum()

172

df.drop\_duplicates(inplace=True)

## balance data operations   
#

loyal\_cust=df[df['Customer Type']=='Loyal Customer']  
disloyal\_cust =df[df['Customer Type']=='disloyal Customer']

loyal\_customer=loyal\_cust.sample(21000)

balace\_df=pd.concat([loyal\_customer,disloyal\_cust],axis=0)  
balace\_df.head()

Gender Customer Type Type of Travel Class \  
30151 Male Loyal Customer Business travel Business   
99825 Male Loyal Customer Business travel Business   
50504 Female Loyal Customer Personal Travel Eco   
39071 Female Loyal Customer Business travel Business   
102382 Male Loyal Customer Business travel Business   
  
 satisfaction Age Flight Distance Inflight entertainment \  
30151 satisfied 46 2105 3   
99825 satisfied 40 631 4   
50504 neutral or dissatisfied 70 1416 1   
39071 satisfied 29 1917 5   
102382 satisfied 47 759 4   
  
 Baggage handling Cleanliness Departure Delay in Minutes \  
30151 3 4 0   
99825 4 4 0   
50504 1 1 0   
39071 4 5 51   
102382 4 3 24   
  
 Arrival Delay in Minutes   
30151 6.0   
99825 0.0   
50504 0.0   
39071 45.0   
102382 19.0

cat\_col=balace\_df.select\_dtypes(include='O')  
cat\_col

Gender Customer Type Type of Travel Class \  
30151 Male Loyal Customer Business travel Business   
99825 Male Loyal Customer Business travel Business   
50504 Female Loyal Customer Personal Travel Eco   
39071 Female Loyal Customer Business travel Business   
102382 Male Loyal Customer Business travel Business   
... ... ... ... ...   
103892 Female disloyal Customer Business travel Business   
103895 Female disloyal Customer Business travel Eco   
103899 Female disloyal Customer Business travel Eco   
103901 Male disloyal Customer Business travel Business   
103902 Female disloyal Customer Business travel Eco   
  
 satisfaction   
30151 satisfied   
99825 satisfied   
50504 neutral or dissatisfied   
39071 satisfied   
102382 satisfied   
... ...   
103892 neutral or dissatisfied   
103895 neutral or dissatisfied   
103899 neutral or dissatisfied   
103901 neutral or dissatisfied   
103902 neutral or dissatisfied   
  
[39905 rows x 5 columns]

num\_col = balace\_df.select\_dtypes(exclude='O')  
num\_col

Age Flight Distance Inflight entertainment Baggage handling \  
30151 46 2105 3 3   
99825 40 631 4 4   
50504 70 1416 1 1   
39071 29 1917 5 4   
102382 47 759 4 4   
... ... ... ... ...   
103892 37 596 3 3   
103895 24 1055 1 5   
103899 23 192 2 4   
103901 30 1995 4 4   
103902 22 1000 1 1   
  
 Cleanliness Departure Delay in Minutes Arrival Delay in Minutes   
30151 4 0 6.0   
99825 4 0 0.0   
50504 1 0 0.0   
39071 5 51 45.0   
102382 3 24 19.0   
... ... ... ...   
103892 3 110 121.0   
103895 1 13 10.0   
103899 2 3 0.0   
103901 4 7 14.0   
103902 1 0 0.0   
  
[39905 rows x 7 columns]

cat\_col

Gender Customer Type Type of Travel Class \  
30151 Male Loyal Customer Business travel Business   
99825 Male Loyal Customer Business travel Business   
50504 Female Loyal Customer Personal Travel Eco   
39071 Female Loyal Customer Business travel Business   
102382 Male Loyal Customer Business travel Business   
... ... ... ... ...   
103892 Female disloyal Customer Business travel Business   
103895 Female disloyal Customer Business travel Eco   
103899 Female disloyal Customer Business travel Eco   
103901 Male disloyal Customer Business travel Business   
103902 Female disloyal Customer Business travel Eco   
  
 satisfaction   
30151 satisfied   
99825 satisfied   
50504 neutral or dissatisfied   
39071 satisfied   
102382 satisfied   
... ...   
103892 neutral or dissatisfied   
103895 neutral or dissatisfied   
103899 neutral or dissatisfied   
103901 neutral or dissatisfied   
103902 neutral or dissatisfied   
  
[39905 rows x 5 columns]

from sklearn.preprocessing import LabelEncoder, OneHotEncoder

encoder= OneHotEncoder(drop='if\_binary')

encoder = OneHotEncoder()

print(OneHotEncoder.\_\_doc\_\_)

Encode categorical features as a one-hot numeric array.  
  
 The input to this transformer should be an array-like of integers or  
 strings, denoting the values taken on by categorical (discrete) features.  
 The features are encoded using a one-hot (aka 'one-of-K' or 'dummy')  
 encoding scheme. This creates a binary column for each category and  
 returns a sparse matrix or dense array (depending on the ``sparse\_output``  
 parameter).  
  
 By default, the encoder derives the categories based on the unique values  
 in each feature. Alternatively, you can also specify the `categories`  
 manually.  
  
 This encoding is needed for feeding categorical data to many scikit-learn  
 estimators, notably linear models and SVMs with the standard kernels.  
  
 Note: a one-hot encoding of y labels should use a LabelBinarizer  
 instead.  
  
 Read more in the :ref:`User Guide <preprocessing\_categorical\_features>`.  
 For a comparison of different encoders, refer to:  
 :ref:`sphx\_glr\_auto\_examples\_preprocessing\_plot\_target\_encoder.py`.  
  
 Parameters  
 ----------  
 categories : 'auto' or a list of array-like, default='auto'  
 Categories (unique values) per feature:  
  
 - 'auto' : Determine categories automatically from the training data.  
 - list : ``categories[i]`` holds the categories expected in the ith  
 column. The passed categories should not mix strings and numeric  
 values within a single feature, and should be sorted in case of  
 numeric values.  
  
 The used categories can be found in the ``categories\_`` attribute.  
  
 .. versionadded:: 0.20  
  
 drop : {'first', 'if\_binary'} or an array-like of shape (n\_features,), default=None  
 Specifies a methodology to use to drop one of the categories per  
 feature. This is useful in situations where perfectly collinear  
 features cause problems, such as when feeding the resulting data  
 into an unregularized linear regression model.  
  
 However, dropping one category breaks the symmetry of the original  
 representation and can therefore induce a bias in downstream models,  
 for instance for penalized linear classification or regression models.  
  
 - None : retain all features (the default).  
 - 'first' : drop the first category in each feature. If only one  
 category is present, the feature will be dropped entirely.  
 - 'if\_binary' : drop the first category in each feature with two  
 categories. Features with 1 or more than 2 categories are  
 left intact.  
 - array : ``drop[i]`` is the category in feature ``X[:, i]`` that  
 should be dropped.  
  
 When `max\_categories` or `min\_frequency` is configured to group  
 infrequent categories, the dropping behavior is handled after the  
 grouping.  
  
 .. versionadded:: 0.21  
 The parameter `drop` was added in 0.21.  
  
 .. versionchanged:: 0.23  
 The option `drop='if\_binary'` was added in 0.23.  
  
 .. versionchanged:: 1.1  
 Support for dropping infrequent categories.  
  
 sparse\_output : bool, default=True  
 When ``True``, it returns a :class:`scipy.sparse.csr\_matrix`,  
 i.e. a sparse matrix in "Compressed Sparse Row" (CSR) format.  
  
 .. versionadded:: 1.2  
 `sparse` was renamed to `sparse\_output`  
  
 dtype : number type, default=np.float64  
 Desired dtype of output.  
  
 handle\_unknown : {'error', 'ignore', 'infrequent\_if\_exist'}, default='error'  
 Specifies the way unknown categories are handled during :meth:`transform`.  
  
 - 'error' : Raise an error if an unknown category is present during transform.  
 - 'ignore' : When an unknown category is encountered during  
 transform, the resulting one-hot encoded columns for this feature  
 will be all zeros. In the inverse transform, an unknown category  
 will be denoted as None.  
 - 'infrequent\_if\_exist' : When an unknown category is encountered  
 during transform, the resulting one-hot encoded columns for this  
 feature will map to the infrequent category if it exists. The  
 infrequent category will be mapped to the last position in the  
 encoding. During inverse transform, an unknown category will be  
 mapped to the category denoted `'infrequent'` if it exists. If the  
 `'infrequent'` category does not exist, then :meth:`transform` and  
 :meth:`inverse\_transform` will handle an unknown category as with  
 `handle\_unknown='ignore'`. Infrequent categories exist based on  
 `min\_frequency` and `max\_categories`. Read more in the  
 :ref:`User Guide <encoder\_infrequent\_categories>`.  
  
 .. versionchanged:: 1.1  
 `'infrequent\_if\_exist'` was added to automatically handle unknown  
 categories and infrequent categories.  
  
 min\_frequency : int or float, default=None  
 Specifies the minimum frequency below which a category will be  
 considered infrequent.  
  
 - If `int`, categories with a smaller cardinality will be considered  
 infrequent.  
  
 - If `float`, categories with a smaller cardinality than  
 `min\_frequency \* n\_samples` will be considered infrequent.  
  
 .. versionadded:: 1.1  
 Read more in the :ref:`User Guide <encoder\_infrequent\_categories>`.  
  
 max\_categories : int, default=None  
 Specifies an upper limit to the number of output features for each input  
 feature when considering infrequent categories. If there are infrequent  
 categories, `max\_categories` includes the category representing the  
 infrequent categories along with the frequent categories. If `None`,  
 there is no limit to the number of output features.  
  
 .. versionadded:: 1.1  
 Read more in the :ref:`User Guide <encoder\_infrequent\_categories>`.  
  
 feature\_name\_combiner : "concat" or callable, default="concat"  
 Callable with signature `def callable(input\_feature, category)` that returns a  
 string. This is used to create feature names to be returned by  
 :meth:`get\_feature\_names\_out`.  
  
 `"concat"` concatenates encoded feature name and category with  
 `feature + "\_" + str(category)`.E.g. feature X with values 1, 6, 7 create  
 feature names `X\_1, X\_6, X\_7`.  
  
 .. versionadded:: 1.3  
  
 Attributes  
 ----------  
 categories\_ : list of arrays  
 The categories of each feature determined during fitting  
 (in order of the features in X and corresponding with the output  
 of ``transform``). This includes the category specified in ``drop``  
 (if any).  
  
 drop\_idx\_ : array of shape (n\_features,)  
 - ``drop\_idx\_[i]`` is the index in ``categories\_[i]`` of the category  
 to be dropped for each feature.  
 - ``drop\_idx\_[i] = None`` if no category is to be dropped from the  
 feature with index ``i``, e.g. when `drop='if\_binary'` and the  
 feature isn't binary.  
 - ``drop\_idx\_ = None`` if all the transformed features will be  
 retained.  
  
 If infrequent categories are enabled by setting `min\_frequency` or  
 `max\_categories` to a non-default value and `drop\_idx[i]` corresponds  
 to a infrequent category, then the entire infrequent category is  
 dropped.  
  
 .. versionchanged:: 0.23  
 Added the possibility to contain `None` values.  
  
 infrequent\_categories\_ : list of ndarray  
 Defined only if infrequent categories are enabled by setting  
 `min\_frequency` or `max\_categories` to a non-default value.  
 `infrequent\_categories\_[i]` are the infrequent categories for feature  
 `i`. If the feature `i` has no infrequent categories  
 `infrequent\_categories\_[i]` is None.  
  
 .. versionadded:: 1.1  
  
 n\_features\_in\_ : int  
 Number of features seen during :term:`fit`.  
  
 .. versionadded:: 1.0  
  
 feature\_names\_in\_ : ndarray of shape (`n\_features\_in\_`,)  
 Names of features seen during :term:`fit`. Defined only when `X`  
 has feature names that are all strings.  
  
 .. versionadded:: 1.0  
  
 feature\_name\_combiner : callable or None  
 Callable with signature `def callable(input\_feature, category)` that returns a  
 string. This is used to create feature names to be returned by  
 :meth:`get\_feature\_names\_out`.  
  
 .. versionadded:: 1.3  
  
 See Also  
 --------  
 OrdinalEncoder : Performs an ordinal (integer)  
 encoding of the categorical features.  
 TargetEncoder : Encodes categorical features using the target.  
 sklearn.feature\_extraction.DictVectorizer : Performs a one-hot encoding of  
 dictionary items (also handles string-valued features).  
 sklearn.feature\_extraction.FeatureHasher : Performs an approximate one-hot  
 encoding of dictionary items or strings.  
 LabelBinarizer : Binarizes labels in a one-vs-all  
 fashion.  
 MultiLabelBinarizer : Transforms between iterable of  
 iterables and a multilabel format, e.g. a (samples x classes) binary  
 matrix indicating the presence of a class label.  
  
 Examples  
 --------  
 Given a dataset with two features, we let the encoder find the unique  
 values per feature and transform the data to a binary one-hot encoding.  
  
 >>> from sklearn.preprocessing import OneHotEncoder  
  
 One can discard categories not seen during `fit`:  
  
 >>> enc = OneHotEncoder(handle\_unknown='ignore')  
 >>> X = [['Male', 1], ['Female', 3], ['Female', 2]]  
 >>> enc.fit(X)  
 OneHotEncoder(handle\_unknown='ignore')  
 >>> enc.categories\_  
 [array(['Female', 'Male'], dtype=object), array([1, 2, 3], dtype=object)]  
 >>> enc.transform([['Female', 1], ['Male', 4]]).toarray()  
 array([[1., 0., 1., 0., 0.],  
 [0., 1., 0., 0., 0.]])  
 >>> enc.inverse\_transform([[0, 1, 1, 0, 0], [0, 0, 0, 1, 0]])  
 array([['Male', 1],  
 [None, 2]], dtype=object)  
 >>> enc.get\_feature\_names\_out(['gender', 'group'])  
 array(['gender\_Female', 'gender\_Male', 'group\_1', 'group\_2', 'group\_3'], ...)  
  
 One can always drop the first column for each feature:  
  
 >>> drop\_enc = OneHotEncoder(drop='first').fit(X)  
 >>> drop\_enc.categories\_  
 [array(['Female', 'Male'], dtype=object), array([1, 2, 3], dtype=object)]  
 >>> drop\_enc.transform([['Female', 1], ['Male', 2]]).toarray()  
 array([[0., 0., 0.],  
 [1., 1., 0.]])  
  
 Or drop a column for feature only having 2 categories:  
  
 >>> drop\_binary\_enc = OneHotEncoder(drop='if\_binary').fit(X)  
 >>> drop\_binary\_enc.transform([['Female', 1], ['Male', 2]]).toarray()  
 array([[0., 1., 0., 0.],  
 [1., 0., 1., 0.]])  
  
 One can change the way feature names are created.  
  
 >>> def custom\_combiner(feature, category):  
 ... return str(feature) + "\_" + type(category).\_\_name\_\_ + "\_" + str(category)  
 >>> custom\_fnames\_enc = OneHotEncoder(feature\_name\_combiner=custom\_combiner).fit(X)  
 >>> custom\_fnames\_enc.get\_feature\_names\_out()  
 array(['x0\_str\_Female', 'x0\_str\_Male', 'x1\_int\_1', 'x1\_int\_2', 'x1\_int\_3'],  
 dtype=object)  
  
 Infrequent categories are enabled by setting `max\_categories` or `min\_frequency`.  
  
 >>> import numpy as np  
 >>> X = np.array([["a"] \* 5 + ["b"] \* 20 + ["c"] \* 10 + ["d"] \* 3], dtype=object).T  
 >>> ohe = OneHotEncoder(max\_categories=3, sparse\_output=False).fit(X)  
 >>> ohe.infrequent\_categories\_  
 [array(['a', 'd'], dtype=object)]  
 >>> ohe.transform([["a"], ["b"]])  
 array([[0., 0., 1.],  
 [1., 0., 0.]])

onehot\_data=encoder.fit\_transform(cat\_col).toarray()

onehot\_data.shape

(39905, 11)

column\_names=list(encoder.get\_feature\_names\_out())  
column\_names

['Gender\_Female',  
 'Gender\_Male',  
 'Customer Type\_Loyal Customer',  
 'Customer Type\_disloyal Customer',  
 'Type of Travel\_Business travel',  
 'Type of Travel\_Personal Travel',  
 'Class\_Business',  
 'Class\_Eco',  
 'Class\_Eco Plus',  
 'satisfaction\_neutral or dissatisfied',  
 'satisfaction\_satisfied']

one\_hot=pd.DataFrame(onehot\_data,columns=column\_names)  
one\_hot

Gender\_Female Gender\_Male Customer Type\_Loyal Customer \  
0 0.0 1.0 1.0   
1 0.0 1.0 1.0   
2 1.0 0.0 1.0   
3 1.0 0.0 1.0   
4 0.0 1.0 1.0   
... ... ... ...   
39900 1.0 0.0 0.0   
39901 1.0 0.0 0.0   
39902 1.0 0.0 0.0   
39903 0.0 1.0 0.0   
39904 1.0 0.0 0.0   
  
 Customer Type\_disloyal Customer Type of Travel\_Business travel \  
0 0.0 1.0   
1 0.0 1.0   
2 0.0 0.0   
3 0.0 1.0   
4 0.0 1.0   
... ... ...   
39900 1.0 1.0   
39901 1.0 1.0   
39902 1.0 1.0   
39903 1.0 1.0   
39904 1.0 1.0   
  
 Type of Travel\_Personal Travel Class\_Business Class\_Eco \  
0 0.0 1.0 0.0   
1 0.0 1.0 0.0   
2 1.0 0.0 1.0   
3 0.0 1.0 0.0   
4 0.0 1.0 0.0   
... ... ... ...   
39900 0.0 1.0 0.0   
39901 0.0 0.0 1.0   
39902 0.0 0.0 1.0   
39903 0.0 1.0 0.0   
39904 0.0 0.0 1.0   
  
 Class\_Eco Plus satisfaction\_neutral or dissatisfied \  
0 0.0 0.0   
1 0.0 0.0   
2 0.0 1.0   
3 0.0 0.0   
4 0.0 0.0   
... ... ...   
39900 0.0 1.0   
39901 0.0 1.0   
39902 0.0 1.0   
39903 0.0 1.0   
39904 0.0 1.0   
  
 satisfaction\_satisfied   
0 1.0   
1 1.0   
2 0.0   
3 1.0   
4 1.0   
... ...   
39900 0.0   
39901 0.0   
39902 0.0   
39903 0.0   
39904 0.0   
  
[39905 rows x 11 columns]

num\_col

Age Flight Distance Inflight entertainment Baggage handling \  
0 46 2105 3 3   
1 40 631 4 4   
2 70 1416 1 1   
3 29 1917 5 4   
4 47 759 4 4   
... ... ... ... ...   
39900 37 596 3 3   
39901 24 1055 1 5   
39902 23 192 2 4   
39903 30 1995 4 4   
39904 22 1000 1 1   
  
 Cleanliness Departure Delay in Minutes Arrival Delay in Minutes   
0 4 0 6.0   
1 4 0 0.0   
2 1 0 0.0   
3 5 51 45.0   
4 3 24 19.0   
... ... ... ...   
39900 3 110 121.0   
39901 1 13 10.0   
39902 2 3 0.0   
39903 4 7 14.0   
39904 1 0 0.0   
  
[39905 rows x 7 columns]

final\_df= pd.concat([one\_hot,num\_col],axis=1)  
final\_df

Gender\_Female Gender\_Male Customer Type\_Loyal Customer \  
0 0.0 1.0 1.0   
1 0.0 1.0 1.0   
2 1.0 0.0 1.0   
3 1.0 0.0 1.0   
4 0.0 1.0 1.0   
... ... ... ...   
39900 1.0 0.0 0.0   
39901 1.0 0.0 0.0   
39902 1.0 0.0 0.0   
39903 0.0 1.0 0.0   
39904 1.0 0.0 0.0   
  
 Customer Type\_disloyal Customer Type of Travel\_Business travel \  
0 0.0 1.0   
1 0.0 1.0   
2 0.0 0.0   
3 0.0 1.0   
4 0.0 1.0   
... ... ...   
39900 1.0 1.0   
39901 1.0 1.0   
39902 1.0 1.0   
39903 1.0 1.0   
39904 1.0 1.0   
  
 Type of Travel\_Personal Travel Class\_Business Class\_Eco \  
0 0.0 1.0 0.0   
1 0.0 1.0 0.0   
2 1.0 0.0 1.0   
3 0.0 1.0 0.0   
4 0.0 1.0 0.0   
... ... ... ...   
39900 0.0 1.0 0.0   
39901 0.0 0.0 1.0   
39902 0.0 0.0 1.0   
39903 0.0 1.0 0.0   
39904 0.0 0.0 1.0   
  
 Class\_Eco Plus satisfaction\_neutral or dissatisfied \  
0 0.0 0.0   
1 0.0 0.0   
2 0.0 1.0   
3 0.0 0.0   
4 0.0 0.0   
... ... ...   
39900 0.0 1.0   
39901 0.0 1.0   
39902 0.0 1.0   
39903 0.0 1.0   
39904 0.0 1.0   
  
 satisfaction\_satisfied Age Flight Distance Inflight entertainment \  
0 1.0 46 2105 3   
1 1.0 40 631 4   
2 0.0 70 1416 1   
3 1.0 29 1917 5   
4 1.0 47 759 4   
... ... ... ... ...   
39900 0.0 37 596 3   
39901 0.0 24 1055 1   
39902 0.0 23 192 2   
39903 0.0 30 1995 4   
39904 0.0 22 1000 1   
  
 Baggage handling Cleanliness Departure Delay in Minutes \  
0 3 4 0   
1 4 4 0   
2 1 1 0   
3 4 5 51   
4 4 3 24   
... ... ... ...   
39900 3 3 110   
39901 5 1 13   
39902 4 2 3   
39903 4 4 7   
39904 1 1 0   
  
 Arrival Delay in Minutes   
0 6.0   
1 0.0   
2 0.0   
3 45.0   
4 19.0   
... ...   
39900 121.0   
39901 10.0   
39902 0.0   
39903 14.0   
39904 0.0   
  
[39905 rows x 18 columns]

num\_col.reset\_index(drop=True,inplace=True)

final\_df

Gender\_Female Gender\_Male Customer Type\_Loyal Customer \  
0 0.0 1.0 1.0   
1 0.0 1.0 1.0   
2 1.0 0.0 1.0   
3 1.0 0.0 1.0   
4 0.0 1.0 1.0   
... ... ... ...   
39900 1.0 0.0 0.0   
39901 1.0 0.0 0.0   
39902 1.0 0.0 0.0   
39903 0.0 1.0 0.0   
39904 1.0 0.0 0.0   
  
 Customer Type\_disloyal Customer Type of Travel\_Business travel \  
0 0.0 1.0   
1 0.0 1.0   
2 0.0 0.0   
3 0.0 1.0   
4 0.0 1.0   
... ... ...   
39900 1.0 1.0   
39901 1.0 1.0   
39902 1.0 1.0   
39903 1.0 1.0   
39904 1.0 1.0   
  
 Type of Travel\_Personal Travel Class\_Business Class\_Eco \  
0 0.0 1.0 0.0   
1 0.0 1.0 0.0   
2 1.0 0.0 1.0   
3 0.0 1.0 0.0   
4 0.0 1.0 0.0   
... ... ... ...   
39900 0.0 1.0 0.0   
39901 0.0 0.0 1.0   
39902 0.0 0.0 1.0   
39903 0.0 1.0 0.0   
39904 0.0 0.0 1.0   
  
 Class\_Eco Plus satisfaction\_neutral or dissatisfied \  
0 0.0 0.0   
1 0.0 0.0   
2 0.0 1.0   
3 0.0 0.0   
4 0.0 0.0   
... ... ...   
39900 0.0 1.0   
39901 0.0 1.0   
39902 0.0 1.0   
39903 0.0 1.0   
39904 0.0 1.0   
  
 satisfaction\_satisfied Age Flight Distance Inflight entertainment \  
0 1.0 46 2105 3   
1 1.0 40 631 4   
2 0.0 70 1416 1   
3 1.0 29 1917 5   
4 1.0 47 759 4   
... ... ... ... ...   
39900 0.0 37 596 3   
39901 0.0 24 1055 1   
39902 0.0 23 192 2   
39903 0.0 30 1995 4   
39904 0.0 22 1000 1   
  
 Baggage handling Cleanliness Departure Delay in Minutes \  
0 3 4 0   
1 4 4 0   
2 1 1 0   
3 4 5 51   
4 4 3 24   
... ... ... ...   
39900 3 3 110   
39901 5 1 13   
39902 4 2 3   
39903 4 4 7   
39904 1 1 0   
  
 Arrival Delay in Minutes   
0 6.0   
1 0.0   
2 0.0   
3 45.0   
4 19.0   
... ...   
39900 121.0   
39901 10.0   
39902 0.0   
39903 14.0   
39904 0.0   
  
[39905 rows x 18 columns]

final\_df.to\_csv('cust\_airline\_cleaned.csv',index=False)

x = final\_df.drop('Customer Type\_disloyal Customer',axis=1)  
y = final\_df[['Customer Type\_disloyal Customer']]

from sklearn.model\_selection import train\_test\_split

x\_train,x\_test ,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.2)

x\_train

Gender\_Female Gender\_Male Customer Type\_Loyal Customer \  
15136 1.0 0.0 1.0   
2017 0.0 1.0 1.0   
24559 0.0 1.0 0.0   
19021 0.0 1.0 1.0   
37652 0.0 1.0 0.0   
... ... ... ...   
16962 1.0 0.0 1.0   
3740 0.0 1.0 1.0   
20783 0.0 1.0 1.0   
20150 0.0 1.0 1.0   
4431 1.0 0.0 1.0   
  
 Type of Travel\_Business travel Type of Travel\_Personal Travel \  
15136 1.0 0.0   
2017 1.0 0.0   
24559 1.0 0.0   
19021 0.0 1.0   
37652 1.0 0.0   
... ... ...   
16962 1.0 0.0   
3740 0.0 1.0   
20783 1.0 0.0   
20150 0.0 1.0   
4431 0.0 1.0   
  
 Class\_Business Class\_Eco Class\_Eco Plus \  
15136 1.0 0.0 0.0   
2017 1.0 0.0 0.0   
24559 0.0 1.0 0.0   
19021 0.0 1.0 0.0   
37652 1.0 0.0 0.0   
... ... ... ...   
16962 1.0 0.0 0.0   
3740 0.0 1.0 0.0   
20783 0.0 1.0 0.0   
20150 0.0 0.0 1.0   
4431 1.0 0.0 0.0   
  
 satisfaction\_neutral or dissatisfied satisfaction\_satisfied Age \  
15136 0.0 1.0 49   
2017 0.0 1.0 70   
24559 1.0 0.0 26   
19021 1.0 0.0 70   
37652 1.0 0.0 38   
... ... ... ...   
16962 1.0 0.0 26   
3740 1.0 0.0 10   
20783 1.0 0.0 43   
20150 1.0 0.0 44   
4431 1.0 0.0 64   
  
 Flight Distance Inflight entertainment Baggage handling Cleanliness \  
15136 189 2 2 4   
2017 3490 4 4 4   
24559 533 4 4 4   
19021 1236 3 4 3   
37652 951 1 5 1   
... ... ... ... ...   
16962 2251 3 3 3   
3740 894 2 3 2   
20783 569 3 4 3   
20150 762 1 4 1   
4431 965 1 1 5   
  
 Departure Delay in Minutes Arrival Delay in Minutes   
15136 5 3.0   
2017 23 25.0   
24559 55 56.0   
19021 87 84.0   
37652 0 0.0   
... ... ...   
16962 11 9.0   
3740 0 13.0   
20783 0 1.0   
20150 0 0.0   
4431 11 4.0   
  
[31924 rows x 17 columns]

y\_train

Customer Type\_disloyal Customer  
15136 0.0  
2017 0.0  
24559 1.0  
19021 0.0  
37652 1.0  
... ...  
16962 0.0  
3740 0.0  
20783 0.0  
20150 0.0  
4431 0.0  
  
[31924 rows x 1 columns]

from sklearn.linear\_model import LogisticRegression

lr = LogisticRegression()

lr.fit(x\_train,y\_train)

c:\Users\Jai\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\utils\validation.py:1300: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().  
 y = column\_or\_1d(y, warn=True)  
c:\Users\Jai\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\linear\_model\\_logistic.py:469: ConvergenceWarning: lbfgs failed to converge (status=1):  
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.  
  
Increase the number of iterations (max\_iter) or scale the data as shown in:  
 https://scikit-learn.org/stable/modules/preprocessing.html  
Please also refer to the documentation for alternative solver options:  
 https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression  
 n\_iter\_i = \_check\_optimize\_result(

LogisticRegression()

lr.score(x\_train,y\_train)

0.9556446560581381

lr.score(x\_test,y\_test)

0.9517604310236812

### for emoji in code press windows key + semicolumn

# to increase accuuracy  
#data cleaning  
# amount of data should be more

# import sklearn.metrics import

df = pd.read\_csv('cust\_airline\_cleaned.csv')  
df

Gender\_Female Gender\_Male Customer Type\_Loyal Customer \  
0 0.0 1.0 1.0   
1 0.0 1.0 1.0   
2 1.0 0.0 1.0   
3 1.0 0.0 1.0   
4 0.0 1.0 1.0   
... ... ... ...   
39900 1.0 0.0 0.0   
39901 1.0 0.0 0.0   
39902 1.0 0.0 0.0   
39903 0.0 1.0 0.0   
39904 1.0 0.0 0.0   
  
 Customer Type\_disloyal Customer Type of Travel\_Business travel \  
0 0.0 1.0   
1 0.0 1.0   
2 0.0 0.0   
3 0.0 1.0   
4 0.0 1.0   
... ... ...   
39900 1.0 1.0   
39901 1.0 1.0   
39902 1.0 1.0   
39903 1.0 1.0   
39904 1.0 1.0   
  
 Type of Travel\_Personal Travel Class\_Business Class\_Eco \  
0 0.0 1.0 0.0   
1 0.0 1.0 0.0   
2 1.0 0.0 1.0   
3 0.0 1.0 0.0   
4 0.0 1.0 0.0   
... ... ... ...   
39900 0.0 1.0 0.0   
39901 0.0 0.0 1.0   
39902 0.0 0.0 1.0   
39903 0.0 1.0 0.0   
39904 0.0 0.0 1.0   
  
 Class\_Eco Plus satisfaction\_neutral or dissatisfied \  
0 0.0 0.0   
1 0.0 0.0   
2 0.0 1.0   
3 0.0 0.0   
4 0.0 0.0   
... ... ...   
39900 0.0 1.0   
39901 0.0 1.0   
39902 0.0 1.0   
39903 0.0 1.0   
39904 0.0 1.0   
  
 satisfaction\_satisfied Age Flight Distance Inflight entertainment \  
0 1.0 46 2105 3   
1 1.0 40 631 4   
2 0.0 70 1416 1   
3 1.0 29 1917 5   
4 1.0 47 759 4   
... ... ... ... ...   
39900 0.0 37 596 3   
39901 0.0 24 1055 1   
39902 0.0 23 192 2   
39903 0.0 30 1995 4   
39904 0.0 22 1000 1   
  
 Baggage handling Cleanliness Departure Delay in Minutes \  
0 3 4 0   
1 4 4 0   
2 1 1 0   
3 4 5 51   
4 4 3 24   
... ... ... ...   
39900 3 3 110   
39901 5 1 13   
39902 4 2 3   
39903 4 4 7   
39904 1 1 0   
  
 Arrival Delay in Minutes   
0 6.0   
1 0.0   
2 0.0   
3 45.0   
4 19.0   
... ...   
39900 121.0   
39901 10.0   
39902 0.0   
39903 14.0   
39904 0.0   
  
[39905 rows x 18 columns]

pred = lr.predict(x\_test)

y\_test['predictin'] = pred

y\_test

Customer Type\_disloyal Customer predictin  
25437 1.0 1.0  
37325 1.0 1.0  
870 0.0 0.0  
2940 0.0 0.0  
35639 1.0 1.0  
... ... ...  
29346 1.0 1.0  
7889 0.0 0.0  
35776 1.0 1.0  
37247 1.0 1.0  
2090 0.0 0.0  
  
[7981 rows x 2 columns]

### MODEL EVALUATION

y\_test['Customer Type\_disloyal Customer'].value\_counts()

Customer Type\_disloyal Customer  
0.0 4207  
1.0 3774  
Name: count, dtype: int64

y\_test['predictin'].value\_counts()

predictin  
0.0 4192  
1.0 3789  
Name: count, dtype: int64

from sklearn.metrics import confusion\_matrix,classification\_report

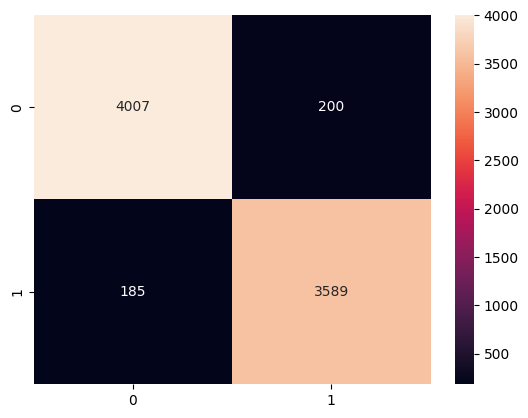
metrix = confusion\_matrix(y\_test['Customer Type\_disloyal Customer'],y\_test['predictin'])  
metrix

array([[4007, 200],  
 [ 185, 3589]], dtype=int64)

import seaborn as sns

sns.heatmap(metrix,annot=True,fmt='d')

<Axes: >



acc = (4007+3589)/(4007+200+185+3589)  
acc

0.9517604310236812

activatin(y) == 0 - 1

predictin\_2 = lr.predict\_proba(x\_test)  
predictin\_2

array([[2.42899391e-03, 9.97571006e-01],  
 [2.68033328e-04, 9.99731967e-01],  
 [9.99475245e-01, 5.24754984e-04],  
 ...,  
 [5.39917952e-03, 9.94600820e-01],  
 [1.57737897e-04, 9.99842262e-01],  
 [9.99988928e-01, 1.10721307e-05]])

print(classification\_report(y\_test['Customer Type\_disloyal Customer'],y\_test['predictin']))

precision recall f1-score support  
  
 0.0 0.96 0.95 0.95 4207  
 1.0 0.95 0.95 0.95 3774  
  
 accuracy 0.95 7981  
 macro avg 0.95 0.95 0.95 7981  
weighted avg 0.95 0.95 0.95 7981