```
import pandas as pd
import numpy as np
```

```
raw_data = pd.read_csv('customer_churn_processed.csv')
data = raw_data.copy()
data
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	Te
0	1	0	1	1	1	1	0	1	0	1	0	
1	0	0	0	1	34	0	1	1	1	0	1	
2	0	0	0	1	2	0	1	1	1	1	0	
3	0	0	0	1	45	1	0	1	1	0	1	
4	1	0	0	1	2	0	1	2	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	
7005	0	0	1	0	24	0	2	1	1	0	1	
7006	1	0	1	0	72	0	2	2	0	1	1	
7007	1	0	1	0	11	1	0	1	1	0	0	
7008	0	1	1	1	4	0	2	2	0	0	0	
7009	0	0	0	1	66	0	1	2	1	0	1	

7010 rows × 20 columns

```
inputs = data.iloc[:,:-1]
target = data.iloc[:,-1]
```

```
inputs
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	Te
0	1	0	1	1	1	1	0	1	0	1	0	
1	0	0	0	1	34	0	1	1	1	0	1	
2	0	0	0	1	2	0	1	1	1	1	0	
3	0	0	0	1	45	1	0	1	1	0	1	
4	1	0	0	1	2	0	1	2	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	
7005	0	0	1	0	24	0	2	1	1	0	1	
7006	1	0	1	0	72	0	2	2	0	1	1	
7007	1	0	1	0	11	1	0	1	1	0	0	
7008	0	1	1	1	4	0	2	2	0	0	0	
7009	0	0	0	1	66	0	1	2	1	0	1	

7010 rows × 19 columns

```
target
```

```
0       0
1       0
2       1
3       0
4       1
       ..
7005    0
7006    0
7007    0
7008    1
7009    0
Name: Churn, Length: 7010, dtype: int64
```

its looks like the targets arent balanced so i need to balance the targets variable

balancing the targets

```
target
```

```
0       0
1       0
2       1
3       0
4       1
       ..
7005    0
7006    0
7007    0
7008    1
7009    0
Name: Churn, Length: 7010, dtype: int64
```

```
#steps is created to balance the data
number_of_churn = int(np.sum(target)) #calculates the total number of churn by sum them up
zero_count = 0
row_to_remove = [] #list to append rows to be removed from data

for i in range(inputs.shape[0]):
  if target[i] == 0:
    zero_count += 1
    if zero_count > number_of_churn:
      row_to_remove.append(i)
```

```
balanced_inputs = np.delete(inputs, row_to_remove, axis=0)
balanced_targets = np.delete(target, row_to_remove, axis=0)
```

```
balanced_targets.sum()/balanced_targets.shape
```

```
array([0.5])
```

```
shuffle_indices = np.arange(balanced_targets.shape[0])
np.random.shuffle(shuffle_indices)

balanced_shuffle_inputs = balanced_inputs[shuffle_indices]
balanced_shuffle_targets = balanced_targets[shuffle_indices]
```

```
from sklearn.preprocessing import StandardScaler
```

```
scale = StandardScaler()
scale.fit(balanced_shuffle_inputs)
```

```
▾ StandardScaler
StandardScaler()
```

```
balanced_scaled_inputs = scale.transform(balanced_shuffle_inputs)
balanced_scaled_inputs
```

```
array([[-0.98344345, -0.49165555,  1.10873429, ..., -1.01123323,
        -0.64696885, -0.78938299],
       [-0.98344345, -0.49165555, -0.90192936, ..., -1.01123323,
         1.50436474,  2.05139755],
       [ 1.01683529, -0.49165555, -0.90192936, ...,  0.71196944,
        -0.84079842, -0.75630875],
       ...,
       [-0.98344345,  2.03394431,  1.10873429, ..., -1.01123323,
         1.19877758,  0.8989407 ],
       [-0.98344345, -0.49165555,  1.10873429, ...,  0.71196944,
         1.06257302,  2.20708285],
       [ 1.01683529, -0.49165555,  1.10873429, ..., -1.01123323,
        -1.50261289, -0.76021064]])
```

```
balanced_scaled_inputs.shape
```

```
(3714, 19)
```

```
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(balanced_scaled_inputs, balanced_shuffle_targets, train_size=0.8, random_state=350)
```

```
x_train.shape, y_train.shape
```

```
((2971, 19), (2971,))
```

```
x_test.shape, y_test.shape
```

```
((743, 19), (743,))
```

```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
```

```
result = LogisticRegression()
```

```
result.fit(x_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
result.score(x_train, y_train)
```

```
0.7744867048131943
```

```
pd.options.display.max_columns = None
pd.options.display.max_rows = None
```

```
model_output = result.predict(x_train)
model_output
```

```
array([0, 1, 0, ..., 1, 1, 1], dtype=int64)
```

finding the **intercept** and **co-efficient**

```
result.intercept_
```

```
array([-0.21650966])
```

```
result.coef_
```

```
array([[ 0.01520306,  0.06921382, -0.01041604,  0.12590718, -1.32129225,
         0.16732179,  0.1648636 ,  1.01649604, -0.16343332, -0.09818304,
        -0.10448801, -0.21323439,  0.29385065,  0.31009114, -0.59889237,
         0.13986171, -0.25159694, -0.63055762,  0.81257605]])
```

```
inputs.columns.values
```

```
array(['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure',
       'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
       'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges',
       'TotalCharges'], dtype=object)
```

```
feature_names = inputs.columns.values
```

```
feature_importance = pd.DataFrame(columns=['featurs Name'], data = feature_names)
feature_importance['Coefficient'] = np.transpose(result.coef_)
feature_importance
```

	featurs Name	Coefficient
0	gender	0.015203
1	SeniorCitizen	0.069214
2	Partner	-0.010416
3	Dependents	0.125907
4	tenure	-1.321292
5	PhoneService	0.167322
6	MultipleLines	0.164864
7	InternetService	1.016496
8	OnlineSecurity	-0.163433
9	OnlineBackup	-0.098183
10	DeviceProtection	-0.104488
11	TechSupport	-0.213234
12	StreamingTV	0.293851
13	StreamingMovies	0.310091
14	Contract	-0.598892
15	PaperlessBilling	0.139862
16	PaymentMethod	-0.251597
17	MonthlyCharges	-0.630558
18	TotalCharges	0.812576

```
feature_importance.index = feature_importance.index + 1

feature_importance.loc[0] = ['Intercept', result.intercept_[0]]

feature_importance.sort_index(inplace=True)

feature_importance
```

	featurs Name	Coefficient
0	Intercept	-0.216510
1	gender	0.015203
2	SeniorCitizen	0.069214
3	Partner	-0.010416
4	Dependents	0.125907
5	tenure	-1.321292
6	PhoneService	0.167322
7	MultipleLines	0.164864
8	InternetService	1.016496
9	OnlineSecurity	-0.163433
10	OnlineBackup	-0.098183
11	DeviceProtection	-0.104488
12	TechSupport	-0.213234
13	StreamingTV	0.293851
14	StreamingMovies	0.310091
15	Contract	-0.598892
16	PaperlessBilling	0.139862
17	PaymentMethod	-0.251597
18	MonthlyCharges	-0.630558
19	TotalCharges	0.812576

interpreting the co-efficient

```
feature_importance['Odds_ratio'] = np.exp(feature_importance.Coefficient)
```

```
feature_importance
```

	featurs Name	Coefficient	Odds_ratio
0	Intercept	-0.216510	0.805325
1	gender	0.015203	1.015319
2	SeniorCitizen	0.069214	1.071665
3	Partner	-0.010416	0.989638
4	Dependents	0.125907	1.134177
5	tenure	-1.321292	0.266790
6	PhoneService	0.167322	1.182135
7	MultipleLines	0.164864	1.179232
8	InternetService	1.016496	2.763495
9	OnlineSecurity	-0.163433	0.849223
10	OnlineBackup	-0.098183	0.906483
11	DeviceProtection	-0.104488	0.900786
12	TechSupport	-0.213234	0.807967
13	StreamingTV	0.293851	1.341584
14	StreamingMovies	0.310091	1.363549
15	Contract	-0.598892	0.549420
16	PaperlessBilling	0.139862	1.150115
17	PaymentMethod	-0.251597	0.777558
18	MonthlyCharges	-0.630558	0.532295
19	TotalCharges	0.812576	2.253706

```
feature_importance.sort_values('Odds_ratio', ascending=False)
```

	featurs Name	Coefficient	Odds_ratio
8	InternetService	1.016496	2.763495
19	TotalCharges	0.812576	2.253706
14	StreamingMovies	0.310091	1.363549
13	StreamingTV	0.293851	1.341584
6	PhoneService	0.167322	1.182135
7	MultipleLines	0.164864	1.179232
16	PaperlessBilling	0.139862	1.150115
4	Dependents	0.125907	1.134177
2	SeniorCitizen	0.069214	1.071665
1	gender	0.015203	1.015319
3	Partner	-0.010416	0.989638
10	OnlineBackup	-0.098183	0.906483
11	DeviceProtection	-0.104488	0.900786
9	OnlineSecurity	-0.163433	0.849223
12	TechSupport	-0.213234	0.807967
0	Intercept	-0.216510	0.805325
17	PaymentMethod	-0.251597	0.777558
15	Contract	-0.598892	0.549420
18	MonthlyCharges	-0.630558	0.532295
5	tenure	-1.321292	0.266790

its clear from the cooefficient that Gender, DeviceProtection, SeniorCitixen, OnlineBackup, Partner are less important in the model as they are farther away from 0 which means it wont really matter if thses variables are taken out

testing the model

```
result.score(x_test, y_test)
```

```
0.7415881561238223
```

```
predictions = result.predict(x_test)
```

```
predicted_probability = result.predict_proba(x_test)
predicted_probability[:,1]
```

```
array([0.78685496, 0.62699275, 0.02166715, 0.34627756, 0.84497837,
       0.85284024, 0.26304713, 0.7913024 , 0.79545238, 0.52752597,
       0.17410627, 0.28626988, 0.83117806, 0.82764841, 0.84132149,
       0.45639689, 0.16647246, 0.81709926, 0.45747586, 0.62448271,
       0.77562843, 0.6179683 , 0.00632968, 0.4423169 , 0.83815424,
       0.87037721, 0.12755158, 0.72745919, 0.82936828, 0.88010184,
       0.05682476, 0.18452307, 0.33767514, 0.65928682, 0.67241996,
       0.54541401, 0.00539792, 0.01208602, 0.65547978, 0.57435331,
       0.27129714, 0.79136936, 0.699646  , 0.64140262, 0.70467799,
       0.84225679, 0.81910514, 0.2475385 , 0.00895502, 0.21427356,
       0.85968377, 0.24778492, 0.73641183, 0.32552791, 0.47150918,
       0.26688541, 0.62170267, 0.57217377, 0.82368662, 0.0863629 ,
       0.06050335, 0.38645867, 0.21100759, 0.61225483, 0.43605068,
       0.04093101, 0.09781588, 0.01198663, 0.04831607, 0.04194908,
       0.80216602, 0.53665865, 0.7415498 , 0.89669039, 0.04391293,
       0.73441422, 0.9067506 , 0.86058741, 0.77205366, 0.00784858,
       0.17848037, 0.21486602, 0.80732891, 0.8302681 , 0.8493353 ,
       0.81826445, 0.83277113, 0.44869408, 0.81630588, 0.47109329,
       0.79868294, 0.781777  , 0.14204394, 0.86514927, 0.78496912,
       0.47174659, 0.00746564, 0.34019419, 0.32891745, 0.20393168,
       0.25614547, 0.17633269, 0.50222163, 0.01426361, 0.37101437,
       0.62868907, 0.10915729, 0.68468244, 0.74515571, 0.21258219,
       0.34433535, 0.12542146, 0.50410005, 0.05492119, 0.76973711,
       0.74445841, 0.70587911, 0.90433665, 0.21662038, 0.77442306,
       0.64712045, 0.15865441, 0.13162601, 0.37955032, 0.77097165,
       0.82258919, 0.86519941, 0.29102278, 0.7708612 , 0.56159732,
       0.03806503, 0.65520975, 0.40249788, 0.79014464, 0.07920876,
       0.05805001, 0.79714484, 0.83344092, 0.72068516, 0.14074302,
       0.24397931, 0.85709642, 0.86082498, 0.6088817 , 0.53083695,
       0.80016661, 0.78420481, 0.88074321, 0.25046204, 0.60141034,
       0.72858893, 0.79825237, 0.07379705, 0.01279418, 0.26651964,
       0.34663898, 0.10853143, 0.30254326, 0.05245183, 0.03959357,
       0.83625589, 0.51461814, 0.86388113, 0.80087614, 0.69911253,
       0.13683517, 0.00992835, 0.58199638, 0.70320938, 0.10335942,
       0.04909849, 0.38862091, 0.11878044, 0.29080701, 0.85458014,
       0.52062415, 0.61836678, 0.77401389, 0.9061932 , 0.08015733,
       0.3510135 , 0.72217496, 0.89341939, 0.15228756, 0.76402648,
       0.67318649, 0.11589204, 0.59000719, 0.08053453, 0.67219785,
       0.8692438 , 0.91540276, 0.01333863, 0.59484789, 0.28523822,
       0.33020642, 0.51757979, 0.81432982, 0.82526433, 0.19991701,
       0.0076901 , 0.59490508, 0.43555346, 0.4462042 , 0.80353593,
       0.68526298, 0.39559872, 0.44822726, 0.33227718, 0.82267207,
       0.89750006, 0.85071823, 0.49705716, 0.32244037, 0.46637476,
       0.10351567, 0.79865495, 0.42662757, 0.0680209 , 0.37363979,
       0.82621355, 0.88586777, 0.31913542, 0.62188908, 0.47396253,
       0.11170335, 0.88125121, 0.84257267, 0.61095273, 0.43563402,
       0.20992855, 0.76475102, 0.77367352, 0.89543717, 0.30714267,
       0.8694924 , 0.58079689, 0.83347209, 0.14525908, 0.58505701,
       0.31820962, 0.44199362, 0.59825327, 0.45881486, 0.89943181,
       0.54326359, 0.89707709, 0.07566504, 0.74047782, 0.89350416,
       0.90119949, 0.3623086 , 0.88362007, 0.40204967, 0.55100118,
       0.04796402, 0.75777723, 0.67455402, 0.3837153 , 0.02551735,
       0.88075011, 0.3776454 , 0.77068868, 0.85519371, 0.8674435 ,
       0.59608233, 0.43288971, 0.12661865, 0.67204039, 0.83817592,
       0.00812529, 0.90041746, 0.74995948, 0.01313725, 0.10339327,
       0.52444808, 0.81896871, 0.68958006, 0.84284354, 0.89821813,
       0.87400164, 0.0588418 , 0.33454671, 0.61290829, 0.54414713,
       0.09662151, 0.09051117, 0.88946915, 0.74542798, 0.02639849,
```

```
x_test
```

```
array([[ 1.01683529, -0.49165555, -0.90192936, ...,  0.71196944,
         0.55442524, -0.10510671],
       [ 1.01683529, -0.49165555, -0.90192936, ...,  0.71196944,
         0.2383608 , -0.38471137],
       [-0.98344345, -0.49165555,  1.10873429, ..., -1.01123323,
        -1.6737417 , -0.31739234],
       ...,
       [-0.98344345, -0.49165555,  1.10873429, ..., -1.01123323,
         0.16501989,  0.43622277],
       [ 1.01683529, -0.49165555, -0.90192936, ..., -0.1496319 ,
        -1.6737417 , -0.90588875],
       [ 1.01683529, -0.49165555, -0.90192936, ..., -1.01123323,
         0.70110707, -0.18895139]])
```

```
predicted = pd.DataFrame(columns = inputs.columns.values, data = x_test)
predicted['Probability'] = predicted_probability[:,1]
predicted['churn'] = predictions
predicted
```