21BCE7371
RADHA KRISHNA GARG

ML LAB ASSIGNMENT-2

LOGISTIC REGRESSION

Consider the employee retention dataset
1. Now do some exploratory data analysis to figure out which variables have a direct and clear impact on employee retention (i.e. whether they leave the company or continue to work)
2. Plot bar charts showing the impact of employee salaries on retention
3. Plot bar charts showing a correlation between department and employee retention
4. Now build a logistic regression model using variables that were narrowed down in step 1
5. Measure the accuracy of the model

CODE

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
df = pd.read_csv("HR_comma_sep.csv")
df.head()
```

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years	Department	salary
0	0.38	0.53	2	157	3	0	1	0	sales	low
1	0.80	0.86	5	262	6	0	1	0	sales	medium
2	0.11	0.88	7	272	4	0	1	0	sales	medium
3	0.72	0.87	5	223	5	0	1	0	sales	low
4	0.37	0.52	2	159	3	0	1	0	sales	low

1. Now do some exploratory data analysis to figure out which variables have a direct and clear impact on employee retention (i.e. whether they leave the company or continue to work)# Data exploration and visualization**

```
left = df[df.left==1]
left.shape
```

```
left = df[df.left==1]
left.shape
```

```
(3571, 10)
```

```
retained = df[df.left==0]
retained.shape
```

```
retained = df[df.left==0]
retained.shape
```

```
(11428, 10)
```

```
df.groupby('left').mean()
```

```
C:\Users\krish\AppData\Local\Temp\ipykernel_27256\588011459.py:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select only columns which should be valid for the function.
  df.groupby('left').mean()
```

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	promotion_last_5years
left							
0	0.666810	0.715473	3.786664	199.060203	3.380032	0.175009	0.026251
1	0.440098	0.718113	3.855503	207.419210	3.876505	0.047326	0.005321

On looking the above data, we can analyze that employee leaving the company mostly depends upon satisfaction_level, average_montly_hours , promotion_last_5years

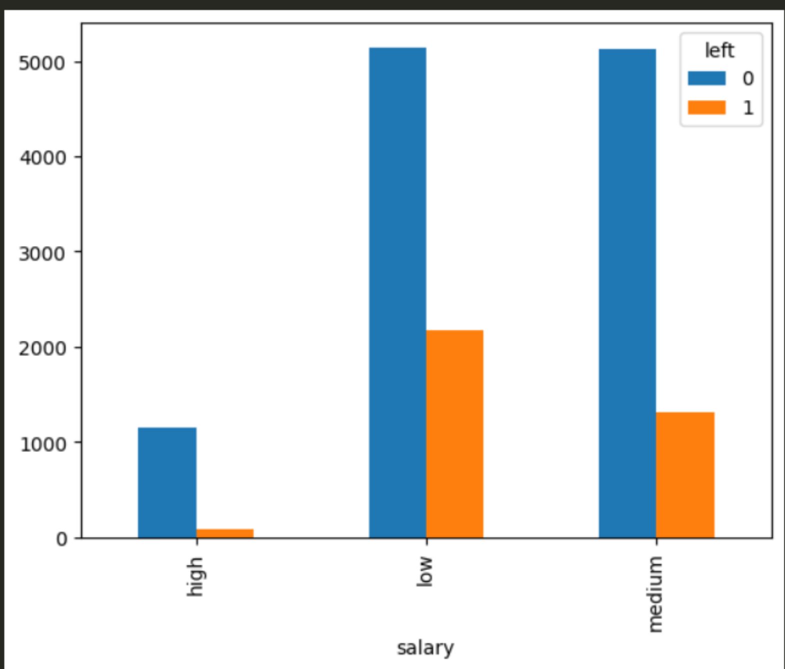
Markdown

On looking the above data, we can analyze that employee leaving the company mostly depends upon satisfaction_level, average_montly_hours , promotion_last_5years

2. Plot bar charts showing impact of employee salaries on retention# Plot bar charts showing impact of employee salaries on retention

```
pd.crosstab(df.salary,df.left).plot(kind='bar')
```

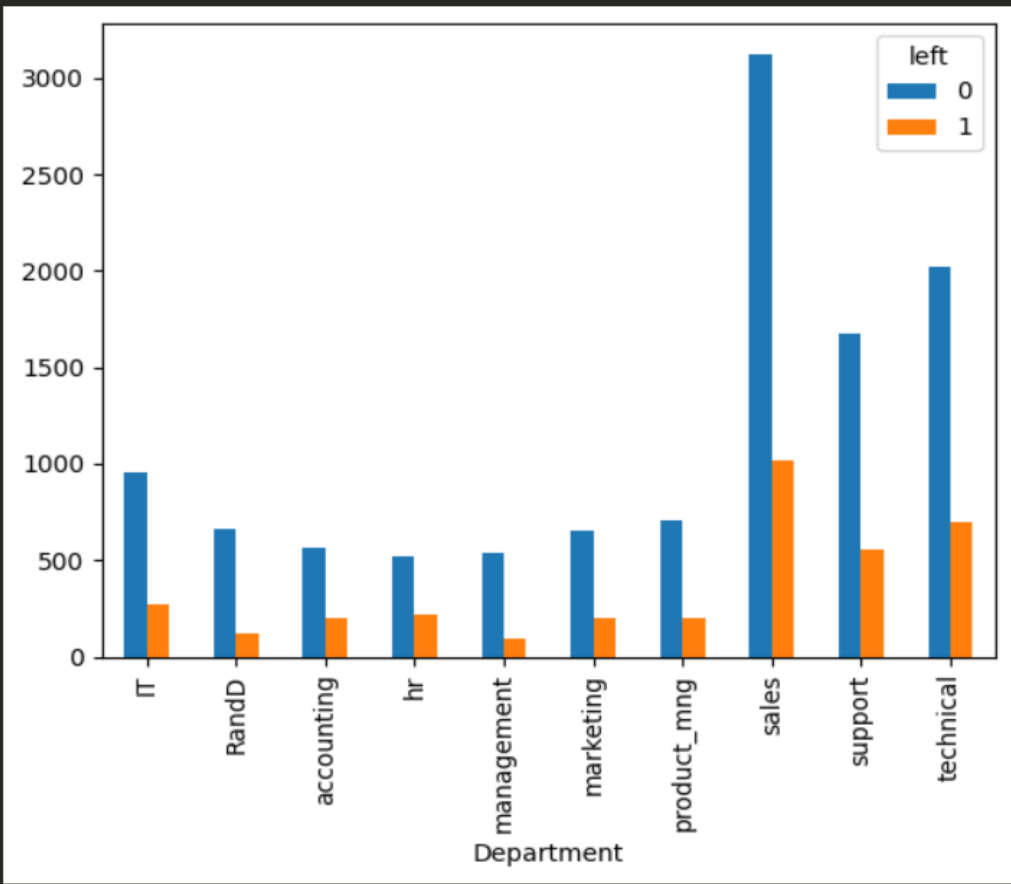
```
&lt;AxesSubplot: xlabel='salary'&gt;
```



3. Plot bar charts showing corelation between department and employee retention

```
pd.crosstab(df.Department,df.left).plot(kind='bar')
```

```
<AxesSubplot: xlabel='Department'>
```



4. Now build logistic regression model using variables that were narrowed down in step 1Now we will create the new data and use it in the model

```
df_new =
df[['satisfaction_level','average_montly_hours','promotion_last_5years'
,'salary']]
df_new.head()
```

	satisfaction_level	average_montly_hours	promotion_last_5years	salary
0	0.38	157	0	low
1	0.80	262	0	medium
2	0.11	272	0	medium
3	0.72	223	0	low
4	0.37	159	0	low

```
# convert categorical data into numerical data
dummy_salary = pd.get_dummies(df_new.salary,prefix='salary')
dummy_salary.head()
```

	salary_high	salary_low	salary_medium
0	0	1	0
1	0	0	1
2	0	0	1
3	0	1	0
4	0	1	0

```
df_new_with_dummy = pd.concat([df_new,dummy_salary],axis='columns')
df_new_with_dummy.head()
```

	satisfaction_level	average_montly_hours	promotion_last_5years	salary	salary_high	salary_low	salary_medium
0	0.38	157	0	low	0	1	0
1	0.80	262	0	medium	0	0	1
2	0.11	272	0	medium	0	0	1
3	0.72	223	0	low	0	1	0
4	0.37	159	0	low	0	1	0

```
df_new_with_dummy.drop('salary',axis='columns',inplace=True)
df_new_with_dummy.head()
```

	satisfaction_level	average_montly_hours	promotion_last_5years	salary_high	salary_low	salary_medium
0	0.38	157	0	0	1	0
1	0.80	262	0	0	0	1
2	0.11	272	0	0	0	1
3	0.72	223	0	0	1	0
4	0.37	159	0	0	1	0

+ Code + Markdown

```
X = df_new_with_dummy
y = df.left
```

```
# split the data
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)
```

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

```
model.fit(X_train,y_train)
```

```
model.fit(X_train,y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
model
predict(X_test)
```

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

5. Measure the accuracy of the model

```
# Model Accuracy
model.score(X_test,y_test)
```

```
0.78
```

ANSWER

0.78

LINEAR REGRESSION

EXERCISE

Predict canada's per capita income in year 2020. There is an exercise folder here on github at same level as this notebook, download that and you will find canada_per_capita_income.csv file. Using this build a regression model and predict the per capita income fo canadian citizens in year 2020

CODE

Importing Important Library

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
%matplotlib inline
```

Reading a csv File

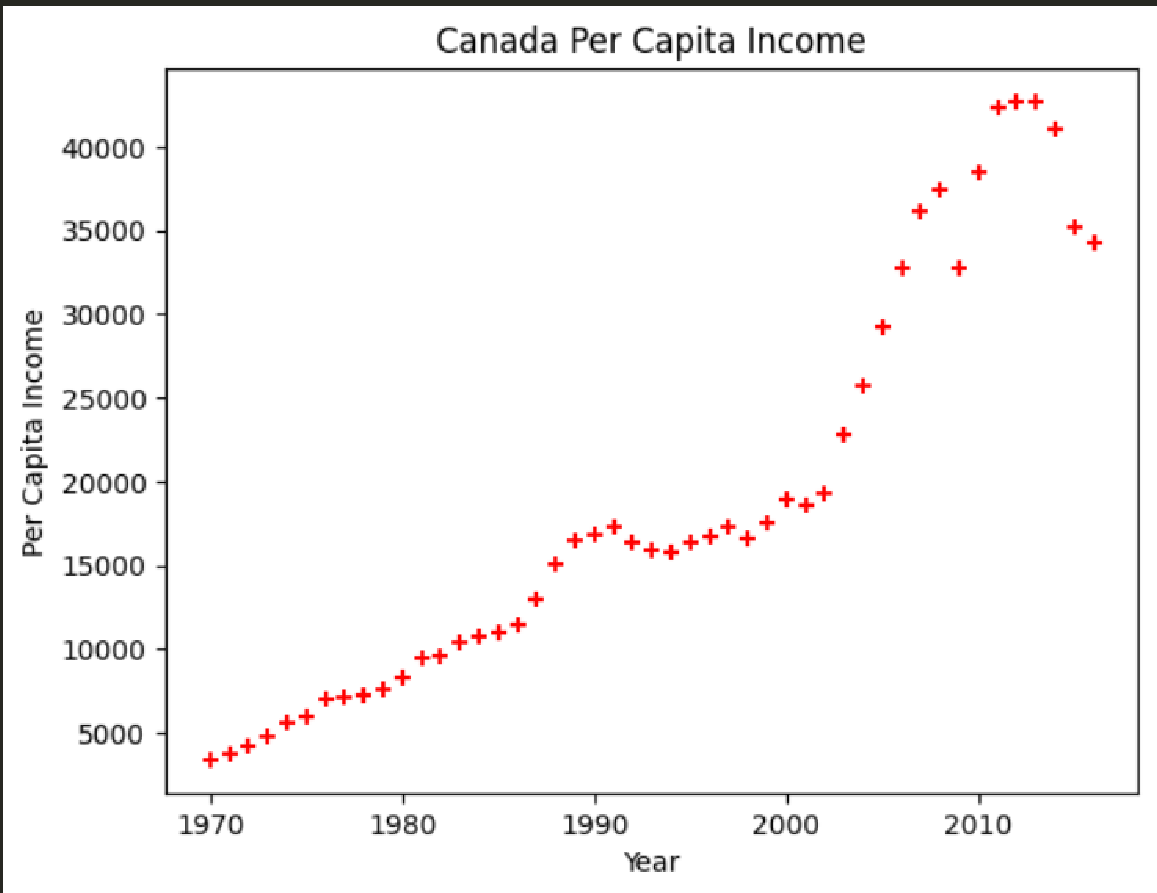
```
df = pd.read_csv("canada_per_capita_income.csv")
df.tail()
```

	year	per capita income (US$)
42	2012	42665.25597
43	2013	42676.46837
44	2014	41039.89360
45	2015	35175.18898
46	2016	34229.19363

Make sure that the data must have a Linear Relationship among Different Variables

```
plt.scatter(df['year'],df['per capita income
(US$)'],color='r',marker='+')
plt.xlabel('Year')
plt.ylabel('Per Capita Income')
plt.title('Canada Per Capita Income')
```

```
Text(0.5, 1.0, 'Canada Per Capita Income')
```



Creating a Linear Regression Model

```
year = df[['year']]
year
df.tail()
```

	year	per capita income (US$)
42	2012	42665.25597
43	2013	42676.46837
44	2014	41039.89360
45	2015	35175.18898
46	2016	34229.19363

Fitting the Model with Available Data

```
income = df['per capita income (US$)']
income
df.tail()
```

	year	per capita income (US$)
42	2012	42665.25597
43	2013	42676.46837
44	2014	41039.89360
45	2015	35175.18898
46	2016	34229.19363

Predicting the Outcome on the basis of Trained Data

```
reg = linear_model.LinearRegression()
reg.fit(year,income)
```

```
▾ LinearRegression
LinearRegression()
```

Checking the Accuracy of Model (0=Least Accurate,1=Most Accurate)

```
reg.predict([[2020]])
```

```
c:\Users\krish\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:420: UserWarning: X does not have valid feature names, but
LinearRegression was fitted with feature names
  warnings.warn(

array([41288.69409442])
```

Checking the Values of Linear Regression Equation

```
reg.coef_  # m
```

```
17]
array([828.46507522])
```

```
reg.intercept_
```

```
18]
-1632210.7578554575
```

```
Y = m * X + b

828.46507522*2020 + -1632210.7578554575
```

```
m = 828.46507522 # reg.coef_ is equal to m
x = 2020 # x is equal to the value we want to predict
b = -1632210.7578554575 # reg.intercept_ is equal to b
y = m*x + b # Equation of Linear Regression
print(y)
```

```
41288.694088942604
```

```
predicted_income = reg.predict(year)
predicted_income
```

```
]
array([ -134.55966672,   693.9054085 ,  1522.37048373,  2350.83555895,
        3179.30063417,  4007.7657094 ,  4836.23078462,  5664.69585984,
        6493.16093506,  7321.62601029,  8150.09108551,  8978.55616073,
        9807.02123595, 10635.48631118, 11463.9513864 , 12292.41646162,
       13120.88153685, 13949.34661207, 14777.81168729, 15606.27676251,
       16434.74183774, 17263.20691296, 18091.67198818, 18920.1370634 ,
       19748.60213863, 20577.06721385, 21405.53228907, 22233.9973643 ,
       23062.46243952, 23890.92751474, 24719.39258996, 25547.85766519,
       26376.32274041, 27204.78781563, 28033.25289085, 28861.71796608,
       29690.1830413 , 30518.64811652, 31347.11319175, 32175.57826697,
       33004.04334219, 33832.50841741, 34660.97349264, 35489.43856786,
       36317.90364308, 37146.3687183 , 37974.83379353])
```

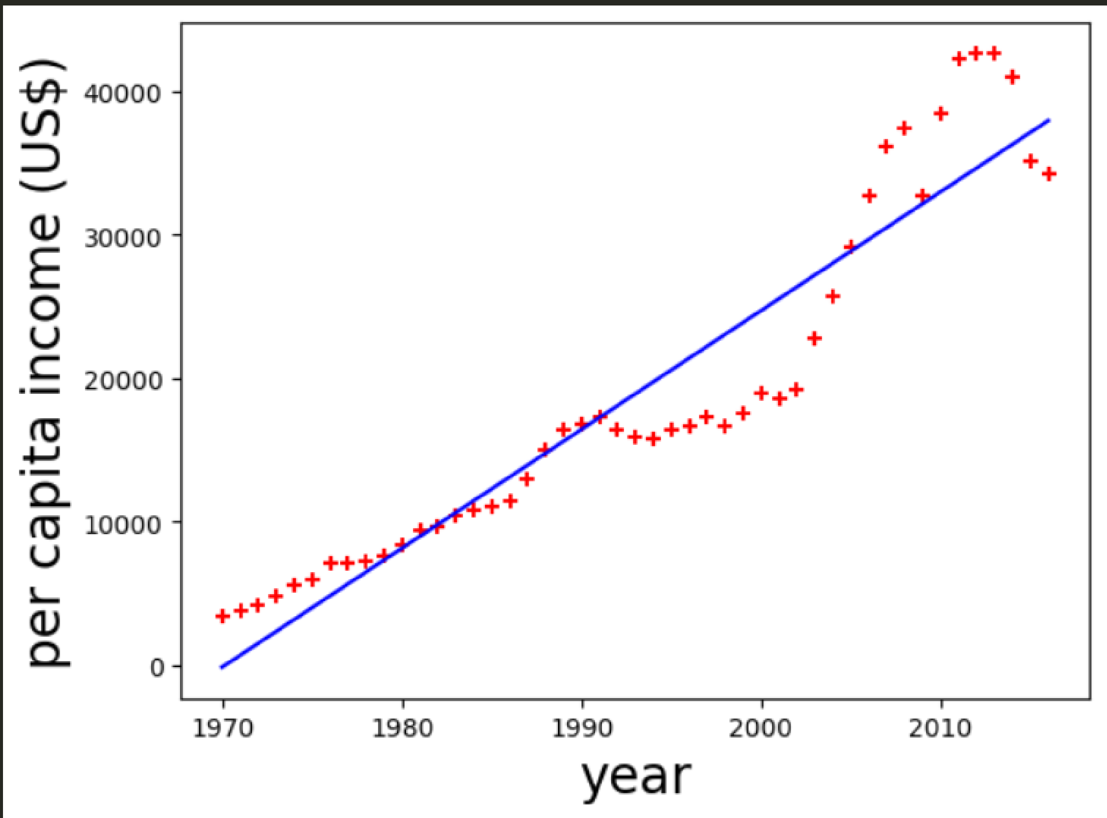
```
predicted_df = pd.DataFrame(year)
predicted_df.head(3)
```

]

	year	predicted income
0	1970	-134.559667
1	1971	693.905409
2	1972	1522.370484

```
%matplotlib inline
plt.xlabel('year', fontsize=20)
plt.ylabel('per capita income (US$)', fontsize=20)
plt.scatter(df['year'],df['per capita income
(US$)'],color='red',marker='+')
plt.plot(predicted_df['year'],predicted_df['predicted
income'],color='blue')
```

```
[<matplotlib.lines.Line2D at 0x26201d2e5c0>]
```



Answer 41288.69409442

Answer
41288.69409442