In [1]:
```
import pandas as pd
import numpy as np
```

Task 1

In [2]:
```
df = pd.read_csv('insurance.csv',index_col=None)
df
```

Out[2]:

	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
...	...	...	...	...	...	...	...
995	39	female	23.275	3	no	northeast	7986.47525
996	39	female	34.100	3	no	southwest	7418.52200
997	63	female	36.850	0	no	southeast	13887.96850
998	33	female	36.290	3	no	northeast	6551.75010
999	36	female	26.885	0	no	northwest	5267.81815

1000 rows × 7 columns

In [3]:
```
df.info
```

Out[3]:
```
<bound method DataFrame.info of      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
..   ...     ...     ...       ...    ...        ...          ...
995   39  female  23.275         3     no  northeast   7986.47525
996   39  female  34.100         3     no  southwest   7418.52200
997   63  female  36.850         0     no  southeast  13887.96850
998   33  female  36.290         3     no  northeast   6551.75010
999   36  female  26.885         0     no  northwest   5267.81815

[1000 rows x 7 columns]>
```

In [4]:
```
from sklearn.preprocessing import LabelEncoder

LE = LabelEncoder()
df['sex'] = LE.fit_transform(df['sex'])
df['smoker'] = LE.fit_transform(df['smoker'])
df['region'] = LE.fit_transform(df['region'])

df
```

Out[4]:

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	1	3	16884.92400
1	18	1	33.770	1	0	2	1725.55230
2	28	1	33.000	3	0	2	4449.46200
3	33	1	22.705	0	0	1	21984.47061
4	32	1	28.880	0	0	1	3866.85520
...	...	...	...	...	...	...	...
995	39	0	23.275	3	0	0	7986.47525
996	39	0	34.100	3	0	3	7418.52200
997	63	0	36.850	0	0	2	13887.96850
998	33	0	36.290	3	0	0	6551.75010
999	36	0	26.885	0	0	1	5267.81815

1000 rows × 7 columns

Label Encoding Info:

sex: Male - 1, Female - 0
smoker: Yes - 1, No - 0
region: southwest - 3, southeast - 2, northwest - 1, northeast - 0

In [5]:
```
data = df.iloc[:,:6].values
data
```

Out[5]:
```
array([[19.   ,  0.   , 27.9  ,  0.   ,  1.   ,  3.   ],
       [18.   ,  1.   , 33.77 ,  1.   ,  0.   ,  2.   ],
       [28.   ,  1.   , 33.   ,  3.   ,  0.   ,  2.   ],
       ...,
       [63.   ,  0.   , 36.85 ,  0.   ,  0.   ,  2.   ],
       [33.   ,  0.   , 36.29 ,  3.   ,  0.   ,  0.   ],
       [36.   ,  0.   , 26.885,  0.   ,  0.   ,  1.   ]])
```

In [6]:
```
charges = df.iloc[:,-1].values
charges
```

Out[6]:
```
array([16884.924   ,  1725.5523  ,  4449.462   , 21984.47061 ,
        3866.8552  ,  3756.6216  ,  8240.5896  ,  7281.5056  ,
        6406.4107  , 28923.13692 ,  2721.3208  , 27808.7251  ,
        1826.843   , 11090.7178  , 39611.7577  ,  1837.237   ,
       10797.3362  ,  2395.17155 , 10602.385   , 36837.467   ,
       13228.84695 ,  4149.736   ,  1137.011   , 37701.8768  ,
        6203.90175 , 14001.1338  , 14451.83515 , 12268.63225 ,
        2775.19215 , 38711.      , 35585.576   ,  2198.18985 ,
        4687.797   , 13770.0979  , 51194.55914 ,  1625.43375 ,
       15612.19335 ,  2302.3     , 39774.2763  , 48173.361   ,
        3046.062   ,  4949.7587  ,  6272.4772  ,  6313.759   ,
        6079.6715  , 20630.28351 ,  3393.35635 ,  3556.9223  ,
       12629.8967  , 38709.176   ,  2211.13075 ,  3579.8287  ,
       23568.272   , 37742.5757  ,  8059.6791  , 47496.49445 ,
       13607.36875 , 34303.1672  , 23244.7902  ,  5989.52365 ,
        8606.2174  ,  4504.6624  , 30166.61817 ,  4133.64165 ,
       14711.7438  ,  1743.214   , 14235.072   ,  6389.37785 ,
        5920.1041  , 17663.1442  , 16577.7795  ,  6799.458   ,
       11741.726   , 11946.6259  ,  7726.854   , 11356.6609  ,
```

In [7]:
```
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
```

In [8]:
```
regr = linear_model.LinearRegression()

regr.fit(data, charges)

y_pred = regr.predict(data)
```

In [9]:
```
print("Coefficients: \n", regr.coef_)
print("Mean squared error: %.2f" % mean_squared_error(charges, y_pr
print("Coefficient of determination: %.2f" % r2_score(charges, y_pr
```

```
Coefficients:
 [  265.00281538  -276.07122609   332.34748636   415.6631237
 23799.08282857  -462.08574687]
Mean squared error: 34925480.46
Coefficient of determination: 0.76
```

Task 2

In [10]:
```
df = pd.read_csv('Fraud.csv',index_col=None)
df
```

Out[10]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nan
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230
...	...	...	...	...	...	...	
1047295	95	PAYMENT	9028.31	C351212668	105818.00	96789.69	M1938
1047296	95	PAYMENT	6290.51	C404585795	96789.69	90499.18	M391
1047297	95	PAYMENT	2499.92	C2079661953	90499.18	87999.25	M227
1047298	95	PAYMENT	4780.32	C2115046838	802.00	0.00	M1154
1047299	95	PAYMENT	789.77	C386614070	30791.00	30001.23	M919

1047300 rows × 11 columns

In [11]:
```
df.info
```

Out[11]:
```
<bound method DataFrame.info of          step      type    amount
nameOrig  oldbalanceOrg  newbalanceOrig  \
0           1   PAYMENT   9839.64  C1231006815      170136.00
160296.36
1           1   PAYMENT   1864.28  C1666544295       21249.00
19384.72
2           1  TRANSFER    181.00  C1305486145         181.00
0.00
3           1  CASH_OUT    181.00   C840083671         181.00
0.00
4           1   PAYMENT  11668.14  C2048537720       41554.00
29885.86
...       ...       ...       ...          ...            ...
...
1047295    95   PAYMENT   9028.31   C351212668      105818.00
96789.69
1047296    95   PAYMENT   6290.51   C404585795       96789.69
90499.18
1047297    95   PAYMENT   2499.92  C2079661953       90499.18
87999.25
1047298    95   PAYMENT   4780.32  C2115046838         802.00
0.00
1047299    95   PAYMENT    789.77   C386614070       30791.00
30001.23

                nameDest  oldbalanceDest  newbalanceDest  isFraud  isF
```

```
            nameDest  oldbalanceDest  newbalanceDest  isFraud  isF
laggedFraud
0        M1979787155             0.0             0.0        0
0
1        M2044282225             0.0             0.0        0
0
2         C553264065             0.0             0.0        1
0
3          C38997010         21182.0             0.0        1
0
4        M1230701703             0.0             0.0        0
0
...              ...             ...             ...      ...
...
1047295  M1938858197             0.0             0.0        0
0
1047296   M391493091             0.0             0.0        0
0
1047297   M227905058             0.0             0.0        0
0
1047298  M1154565434             0.0             0.0        0
0
1047299   M919485291             0.0             0.0        0
0

[1047300 rows x 11 columns]>
```

In [12]:
```
from sklearn.preprocessing import LabelEncoder

LE = LabelEncoder()
df['type'] = LE.fit_transform(df['type'])
df['nameOrig'] = LE.fit_transform(df['nameOrig'])
df['nameDest'] = LE.fit_transform(df['nameDest'])

df
```

Out[12]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldb
0	1	3	9839.64	125004	170136.00	160296.36	274496	
1	1	3	1864.28	360882	21249.00	19384.72	286268	
2	1	4	181.00	165023	181.00	0.00	73503	
3	1	1	181.00	960498	181.00	0.00	65423	
4	1	3	11668.14	567221	41554.00	29885.86	137860	
...	...	...	...	...	...	...	...	
1047295	95	3	9028.31	694724	105818.00	96789.69	266884	
1047296	95	3	6290.51	723239	96789.69	90499.18	337296	
1047297	95	3	2499.92	583942	90499.18	87999.25	307566	
1047298	95	3	4780.32	603263	802.00	0.00	124109	
1047299	95	3	789.77	713704	30791.00	30001.23	433968	

1047300 rows × 11 columns

In [13]:
```
data = df.iloc[:,:10].values
data
```

Out[13]:
```
array([[1.00000e+00, 3.00000e+00, 9.83964e+03, ..., 0.00000e+00,
        0.00000e+00, 0.00000e+00],
       [1.00000e+00, 3.00000e+00, 1.86428e+03, ..., 0.00000e+00,
        0.00000e+00, 0.00000e+00],
       [1.00000e+00, 4.00000e+00, 1.81000e+02, ..., 0.00000e+00,
        0.00000e+00, 1.00000e+00],
       ...,
       [9.50000e+01, 3.00000e+00, 2.49992e+03, ..., 0.00000e+00,
        0.00000e+00, 0.00000e+00],
       [9.50000e+01, 3.00000e+00, 4.78032e+03, ..., 0.00000e+00,
        0.00000e+00, 0.00000e+00],
       [9.50000e+01, 3.00000e+00, 7.89770e+02, ..., 0.00000e+00,
        0.00000e+00, 0.00000e+00]])
```

In [14]:
```
label = df.iloc[:,-2].values
label
```

Out[14]:
```
array([0, 0, 1, ..., 0, 0, 0])
```

In [15]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(data, label, te
```

In [16]:
```
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression(random_state=16)

logreg.fit(X_train, y_train)
```

Out[16]:
```
LogisticRegression(random_state=16)
```

In [17]:
```
y_pred = logreg.predict(X_test)
```

In [18]:
```
from sklearn import metrics

cm = metrics.confusion_matrix(y_test, y_pred)
cm
```

Out[18]:
```
array([[209217,     22],
       [   138,     83]])
```

In [19]:
```
from sklearn.metrics import classification_report
target_names = ['not spam', 'spam']
print(classification_report(y_test, y_pred, target_names=target_nam
```

```
              precision    recall  f1-score   support

    not spam       1.00      1.00      1.00    209239
        spam       0.79      0.38      0.51       221

    accuracy                           1.00    209460
   macro avg       0.89      0.69      0.75    209460
weighted avg       1.00      1.00      1.00    209460
```

In [20]:
```
acc = metrics.accuracy_score(y_test, y_pred)
print("Accuracy: "+str(acc))
```

```
Accuracy: 0.9992361310035329
```