Major Project

Project Name: Data Science June Major Project

GROUP CODE : DS-06-BSP3

Group Members:

- Rewa Abhyankar
- Kharankumar Raju
- Vyshnavi Krishna
- Amogh Sadvelkar
- Omar Inamdar
- Ayush Dubey
- Dhanush Nani
- Samad Shaikh

Project Description:

Problem statement: Create a classification model to predict whether a person makes over $50k a Year.

Context:

This data was extracted from the 1994 Census bureau database by Ronny Kohavi and Barry Becker (Data Mining and Visualization, Silicon Graphics).

Dataset:

https://drive.google.com/file/d/1E_IaMMGqP8qDA3O9VW1rzhrXeaq2dY1S/view?usp=sharing.

Steps to consider:

1) Rename the columns.

2) Remove handle null values (if any).

3) Split data into training and test data.

4) Apply the following models on the training dataset and generate the predicted value for the
test dataset
a. Decision Tree
b. Random Forest Classifier
c. Logistic Regression
d. KNN Classifier
e. SVC Classifier (with linear kernel)

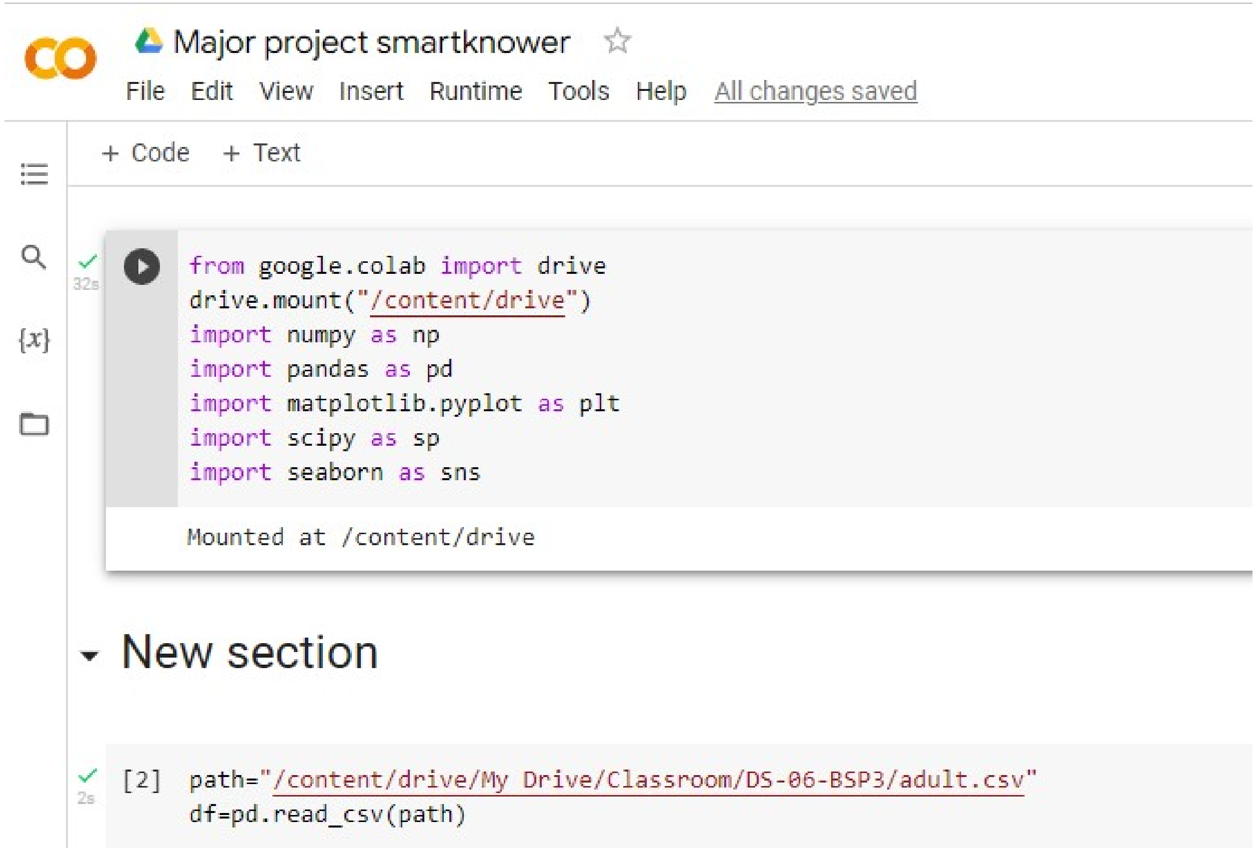
5) Predict the income for test data

6) Compute Confusion matrix and classification report for each of these models.

7) Validate the result for Precision, Recall, F1-score and Accuracy for each model based on values from confusion_matrix and classification_report

8) Generate the percentage of misclassification in each of these models.

9) Report the model with the best accuracy.



Details of features:

The columns are described as follows:

1) Age
2) Workclass
3) Fnlwgt
4) Education
5) education_num
6) marital_status
7) occupation
8) relationship

9) race
10) sex
11) capital_gain
12) capital_loss
13) hours_per_week
14) native_country
15) income

1) Rename the columns.

```
df.columns =["Age","Workclass","Fnlwgt","Education","education_num","marital_status","occupation","relationship","race","sex","capital_gain","capital_loss","hours_per_week","native_country","income"]
```

```
df
```

	Age	Workclass	Fnlwgt	Education	education_num	marital_status	occupation	relationship	race	sex	capital_gain	capital_loss	hours_per_week
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40
...	...	...	...	...	...	...	...	...	...	...	...	...	...
32556	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	38
32557	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40
32558	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	40
32559	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	0	20
32560	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	0	40

32561 rows × 15 columns

```
[3] df.head()
```

	Age	Workclass	Fnlwgt	Education	education_num	marital_status	occupation	relationship	race	sex	capital_gain	capital_loss	hours_per_week	native_country	income
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

2) Remove handle null values (if any).

```
df['native_country'].value_counts()/len(df)*100
```

```
United-States                 89.585701
Mexico                         1.974755
?                              1.790486
Philippines                    0.608089
Germany                        0.420749
Canada                         0.371610
Puerto-Rico                    0.350112
El-Salvador                    0.325543
India                          0.307116
Cuba                           0.291760
England                        0.276404
Jamaica                        0.248764
South                          0.245693
China                          0.230337
Italy                          0.224195
Dominican-Republic             0.214981
Vietnam                        0.205768
Guatemala                      0.196554
Japan                          0.190412
Poland                         0.184270
Columbia                       0.181198
Taiwan                         0.156629
Haiti                          0.135131
Iran                           0.132060
Portugal                       0.113633
Nicaragua                      0.104419
Peru                           0.095206
France                         0.089064
Greece                         0.089064
Ecuador                        0.085992
Ireland                        0.073708
Hong                           0.061423
Cambodia                       0.058352
Trinadad&Tobago                0.058352
Laos                           0.055281
Thailand                       0.055281
Yugoslavia                     0.049139
Outlying-US(Guam-USVI-etc)     0.042996
Honduras                       0.039925
Hungary                        0.039925
Scotland                       0.036854
Holand-Netherlands             0.003071
Name: native_country, dtype: float64
```

2) Remove handle null values (if any).

```
df.native_country.replace(' ?',' United-States',inplace=True)
```

```
df.native_country.value_counts()
```

```
 United-States                 29753
 Mexico                          643
 Philippines                     198
 Germany                         137
 Canada                          121
 Puerto-Rico                     114
 El-Salvador                     106
 India                           100
 Cuba                             95
 England                          90
 Jamaica                          81
 South                            80
 China                            75
 Italy                            73
 Dominican-Republic               70
 Vietnam                          67
 Guatemala                        64
 Japan                            62
 Poland                           60
 Columbia                         59
 Taiwan                           51
 Haiti                            44
 Iran                             43
 Portugal                         37
 Nicaragua                        34
 Peru                             31
 France                           29
 Greece                           29
 Ecuador                          28
 Ireland                          24
 Hong                             20
 Cambodia                         19
 Trinadad&Tobago                  19
 Laos                             18
 Thailand                         18
 Yugoslavia                       16
 Outlying-US(Guam-USVI-etc)       14
 Honduras                         13
 Hungary                          13
 Scotland                         12
 Holand-Netherlands                1
Name: native_country, dtype: int64
```

2) Remove handle null values (if any).

```
df['occupation'].value_counts()
```

```
 Prof-specialty       4140
 Craft-repair         4099
 Exec-managerial      4066
 Adm-clerical         3770
 Sales                3650
 Other-service        3295
 Machine-op-inspct    2002
 ?                    1843
 Transport-moving     1597
 Handlers-cleaners    1370
 Farming-fishing       994
 Tech-support          928
 Protective-serv       649
 Priv-house-serv       149
 Armed-Forces            9
Name: occupation, dtype: int64
```

```
df.occupation.replace(' ?',' Prof-specialty',inplace=True)
```

```
df['occupation'].value_counts()
```

```
 Prof-specialty       5983
 Craft-repair         4099
 Exec-managerial      4066
 Adm-clerical         3770
 Sales                3650
 Other-service        3295
 Machine-op-inspct    2002
 Transport-moving     1597
 Handlers-cleaners    1370
 Farming-fishing       994
 Tech-support          928
 Protective-serv       649
 Priv-house-serv       149
 Armed-Forces            9
Name: occupation, dtype: int64
```

2) Remove handle null values (if any).

```
df.isin([' ?']).sum(axis=0)
```

```
Age                   0
Workclass          1836
Fnlwgt                0
Education             0
education_num         0
marital_status        0
occupation            0
relationship          0
race                  0
sex                   0
capital_gain          0
capital_loss          0
hours_per_week        0
native_country        0
income                0
dtype: int64
```

```
df['Workclass'].value_counts()
```

```
 Private             22696
 Self-emp-not-inc     2541
 Local-gov            2093
 ?                    1836
 State-gov            1298
 Self-emp-inc         1116
 Federal-gov           960
 Without-pay            14
 Never-worked            7
Name: Workclass, dtype: int64
```

```
df.Workclass.replace(' ?',' Private',inplace=True)
```

```
df.isin([' ?']).sum(axis=0)
```

```
Age               0
Workclass         0
Fnlwgt            0
Education         0
education_num     0
marital_status    0
occupation        0
relationship      0
race              0
sex               0
capital_gain      0
capital_loss      0
hours_per_week    0
native_country    0
income            0
dtype: int64
```

```
df.columns
```

```
Index(['Age', 'Workclass', 'Fnlwgt', 'Education', 'education_num',
       'marital_status', 'occupation', 'relationship', 'race', 'sex',
       'capital_gain', 'capital_loss', 'hours_per_week', 'native_country',
       'income'],
      dtype='object')
```

2) Remove handle null values (if any).

```
df.isnull().sum()
```

```
Age               0
Workclass         0
Fnlwgt            0
Education         0
education_num     0
marital_status    0
occupation        0
relationship      0
race              0
sex               0
capital_gain      0
capital_loss      0
hours_per_week    0
native_country    0
income            0
dtype: int64
```

```
df.dtypes
```

```
Age                int64
Workclass         object
Fnlwgt             int64
Education         object
education_num      int64
marital_status    object
occupation        object
relationship      object
race              object
sex               object
capital_gain       int64
capital_loss       int64
hours_per_week     int64
native_country    object
income            object
dtype: object
```

2) Remove handle null values (if any).

```
df.Age.unique()
```

```
array([39, 50, 38, 53, 28, 37, 49, 52, 31, 42, 30, 23, 32, 40, 34, 25, 43,
       54, 35, 59, 56, 19, 20, 45, 22, 48, 21, 24, 57, 44, 41, 29, 18, 47,
       46, 36, 79, 27, 67, 33, 76, 17, 55, 61, 70, 64, 71, 68, 66, 51, 58,
       26, 60, 90, 75, 65, 77, 62, 63, 80, 72, 74, 69, 73, 81, 78, 88, 82,
       83, 84, 85, 86, 87], dtype=int64)
```

```
df.Workclass.unique()
```

```
array([' State-gov', ' Self-emp-not-inc', ' Private', ' Federal-gov',
       ' Local-gov', ' Self-emp-inc', ' Without-pay', ' Never-worked'],
      dtype=object)
```

```
df.Fnlwgt.unique()
```

```
array([ 77516,  83311, 215646, ...,  34066,  84661, 257302], dtype=int64)
```

```
df.Education.unique()
```

```
array([' Bachelors', ' HS-grad', ' 11th', ' Masters', ' 9th',
       ' Some-college', ' Assoc-acdm', ' Assoc-voc', ' 7th-8th',
       ' Doctorate', ' Prof-school', ' 5th-6th', ' 10th', ' 1st-4th',
       ' Preschool', ' 12th'], dtype=object)
```

```
df.education_num.unique()
```

```
array([13,  9,  7, 14,  5, 10, 12, 11,  4, 16, 15,  3,  6,  2,  1,  8],
      dtype=int64)
```

```
df.marital_status.unique()
```

```
array([' Never-married', ' Married-civ-spouse', ' Divorced',
       ' Married-spouse-absent', ' Separated', ' Married-AF-spouse',
       ' Widowed'], dtype=object)
```

```
df.occupation.unique()
```

```
array([' Adm-clerical', ' Exec-managerial', ' Handlers-cleaners',
       ' Prof-specialty', ' Other-service', ' Sales', ' Craft-repair',
       ' Transport-moving', ' Farming-fishing', ' Machine-op-inspct',
       ' Tech-support', ' Protective-serv', ' Armed-Forces',
```

2) Remove handle null values (if any).

```
df.relationship.unique()
```

```
array([' Not-in-family', ' Husband', ' Wife', ' Own-child', ' Unmarried',
       ' Other-relative'], dtype=object)
```

```
df.race.unique()
```

```
array([' White', ' Black', ' Asian-Pac-Islander', ' Amer-Indian-Eskimo',
       ' Other'], dtype=object)
```

```
df.sex.unique()
```

```
array([' Male', ' Female'], dtype=object)
```

```
df.capital_gain.unique()
```

```
array([ 2174,     0, 14084,  5178,  5013,  2407, 14344, 15024,  7688,
       34095,  4064,  4386,  7298,  1409,  3674,  1055,  3464,  2050,
        2176,   594, 20051,  6849,  4101,  1111,  8614,  3411,  2597,
       25236,  4650,  9386,  2463,  3103, 10605,  2964,  3325,  2580,
        3471,  4865, 99999,  6514,  1471,  2329,  2105,  2885, 25124,
       10520,  2202,  2961, 27828,  6767,  2228,  1506, 13550,  2635,
        5556,  4787,  3781,  3137,  3818,  3942,   914,   401,  2829,
        2977,  4934,  2062,  2354,  5455, 15020,  1424,  3273, 22040,
        4416,  3908, 10566,   991,  4931,  1086,  7430,  6497,   114,
        7896,  2346,  3418,  3432,  2907,  1151,  2414,  2290, 15831,
       41310,  4508,  2538,  3456,  6418,  1848,  3887,  5721,  9562,
        1455,  2036,  1831, 11678,  2936,  2993,  7443,  6360,  1797,
        1173,  4687,  6723,  2009,  6097,  2653,  1639, 18481,  7978,
        2387,  5060], dtype=int64)
```

```
df.capital_loss.unique()
```

```
array([   0, 2042, 1408, 1902, 1573, 1887, 1719, 1762, 1564, 2179, 1816,
       1980, 1977, 1876, 1340, 2206, 1741, 1485, 2339, 2415, 1380, 1721,
       2051, 2377, 1669, 2352, 1672,  653, 2392, 1504, 2001, 1590, 1651,
       1628, 1848, 1740, 2002, 1579, 2258, 1602,  419, 2547, 2174, 2205,
       1726, 2444, 1138, 2238,  625,  213, 1539,  880, 1668, 1092, 1594,
       3004, 2231, 1844,  810, 2824, 2559, 2057, 1974,  974, 2149, 1825,
       1735, 1258, 2129, 2603, 2282,  323, 4356, 2246, 1617, 1648, 2489,
       3770, 1755, 3683, 2267, 2080, 2457,  155, 3900, 2201, 1944, 2467,
       2163, 2754, 2472, 1411], dtype=int64)
```

2) Remove handle null values (if any).

```
df.hours_per_week.unique()
```

```
array([40, 13, 16, 45, 50, 80, 30, 35, 60, 20, 52, 44, 15, 25, 38, 43, 55,
       48, 58, 32, 70,  2, 22, 56, 41, 28, 36, 24, 46, 42, 12, 65,  1, 10,
       34, 75, 98, 33, 54,  8,  6, 64, 19, 18, 72,  5,  9, 47, 37, 21, 26,
       14,  4, 59,  7, 99, 53, 39, 62, 57, 78, 90, 66, 11, 49, 84,  3, 17,
       68, 27, 85, 31, 51, 77, 63, 23, 87, 88, 73, 89, 97, 94, 29, 96, 67,
       82, 86, 91, 81, 76, 92, 61, 74, 95], dtype=int64)
```

```
df.native_country.unique()
```

```
array([' United-States', ' Cuba', ' Jamaica', ' India', ' Mexico',
       ' South', ' Puerto-Rico', ' Honduras', ' England', ' Canada',
       ' Germany', ' Iran', ' Philippines', ' Italy', ' Poland',
       ' Columbia', ' Cambodia', ' Thailand', ' Ecuador', ' Laos',
       ' Taiwan', ' Haiti', ' Portugal', ' Dominican-Republic',
       ' El-Salvador', ' France', ' Guatemala', ' China', ' Japan',
       ' Yugoslavia', ' Peru', ' Outlying-US(Guam-USVI-etc)', ' Scotland',
       ' Trinadad&Tobago', ' Greece', ' Nicaragua', ' Vietnam', ' Hong',
       ' Ireland', ' Hungary', ' Holand-Netherlands'], dtype=object)
```

```
df.income.unique()
```

```
array([' <=50K', ' >50K'], dtype=object)
```

```
df.dtypes
```

```
Age                int64
Workclass         object
Fnlwgt             int64
Education         object
education_num      int64
marital_status    object
occupation        object
relationship      object
race              object
sex               object
capital_gain       int64
capital_loss       int64
hours_per_week     int64
native_country    object
income            object
dtype: object
```

ENCODING DATA INTO CATEGORICAL TO NUMERICAL

```python
label_encoding = {'Education':{' Bachelors':9,' HS-grad':8,' 11th':6,' Masters':10,' 9th':4,
       ' Some-college':11, ' Assoc-acdm':12, ' Assoc-voc':13, ' 7th-8th':3,
       ' Doctorate':14, ' Prof-school':15, ' 5th-6th':2,' 10th':5,' 1st-4th':1,
       ' Preschool':0,' 12th':7},'income':{' <=50K':0,' >50K':1}}
```

```python
df =df.replace(label_encoding)
df
```

	Age	Workclass	Fnlwgt	Education	education_num	marital_status	occupation	relationship	race	sex	capital_gain	capital_loss	hours_per_week
0	39	State-gov	77516	9	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40
1	50	Self-emp-not-inc	83311	9	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13
2	38	Private	215646	8	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40
3	53	Private	234721	6	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40
4	28	Private	338409	9	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40
...	...	...	...	...	...	...	...	...	...	...	...	...	...
32556	27	Private	257302	12	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	38
32557	40	Private	154374	8	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40
32558	58	Private	151910	8	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	40
32559	22	Private	201490	8	9	Never-married	Adm-clerical	Own-child	White	Male	0	0	20
32560	52	Self-emp-inc	287927	8	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	0	40

32561 rows × 15 columns

```
dummies = pd.get_dummies(df.Workclass)
merge = pd.concat([df,dummies],axis=1)
merge
```

	Age	Workclass	Fnlwgt	Education	education_num	marital_status	occupation	relationship	race	sex	...	native_country	income	Federal-gov	Local gov
0	39	State-gov	77516	9	13	Never-married	Adm-clerical	Not-in-family	White	Male	...	United-States	0	0	(
1	50	Self-emp-not-inc	83311	9	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	...	United-States	0	0	(
2	38	Private	215646	8	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	...	United-States	0	0	(
3	53	Private	234721	6	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	...	United-States	0	0	(
4	28	Private	338409	9	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	...	Cuba	0	0	(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
32556	27	Private	257302	12	12	Married-civ-spouse	Tech-support	Wife	White	Female	...	United-States	0	0	(
32557	40	Private	154374	8	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	...	United-States	1	0	(
32558	58	Private	151910	8	9	Widowed	Adm-clerical	Unmarried	White	Female	...	United-States	0	0	(
32559	22	Private	201490	8	9	Never-married	Adm-clerical	Own-child	White	Male	...	United-States	0	0	(
32560	52	Self-emp-inc	287927	8	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	...	United-States	1	0	(

32561 rows × 23 columns

```
final = merge.drop(['Workclass'],axis =1)
final
```

	Age	Fnlwgt	Education	education_num	marital_status	occupation	relationship	race	sex	capital_gain	...	native_country	income	Federal-gov	Loc g
0	39	77516	9	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	...	United-States	0	0	
1	50	83311	9	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	...	United-States	0	0	
2	38	215646	8	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	...	United-States	0	0	
3	53	234721	6	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	...	United-States	0	0	
4	28	338409	9	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	...	Cuba	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
32556	27	257302	12	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	...	United-States	0	0	
32557	40	154374	8	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	...	United-States	1	0	
32558	58	151910	8	9	Widowed	Adm-clerical	Unmarried	White	Female	0	...	United-States	0	0	
32559	22	201490	8	9	Never-married	Adm-clerical	Own-child	White	Male	0	...	United-States	0	0	
32560	52	287927	8	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	...	United-States	1	0	

32561 rows × 22 columns

```
dummies = pd.get_dummies(final.marital_status)
merg = pd.concat([final,dummies],axis=1)
merg
```

	Age	Fnlwgt	Education	education_num	marital_status	occupation	relationship	race	sex	capital_gain	...	Self-emp-not-inc	State-gov	Without-pay	Divorced	Mar spo
0	39	77516	9	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	...	0	1	0	0	
1	50	83311	9	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	...	1	0	0	0	
2	38	215646	8	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	...	0	0	0	1	
3	53	234721	6	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	...	0	0	0	0	
4	28	338409	9	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	...	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
32556	27	257302	12	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	...	0	0	0	0	
32557	40	154374	8	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	...	0	0	0	0	
32558	58	151910	8	9	Widowed	Adm-clerical	Unmarried	White	Female	0	...	0	0	0	0	
32559	22	201490	8	9	Never-married	Adm-clerical	Own-child	White	Male	0	...	0	0	0	0	
32560	52	287927	8	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	...	0	0	0	0	

32561 rows × 29 columns

```
mer = merg.drop(['marital_status'],axis =1)
mer
```

	Age	Fnlwgt	Education	education_num	occupation	relationship	race	sex	capital_gain	capital_loss	...	Self-emp-not-inc	State-gov	Without-pay	Divorced	Marrie A spou
0	39	77516	9	13	Adm-clerical	Not-in-family	White	Male	2174	0	...	0	1	0	0	
1	50	83311	9	13	Exec-managerial	Husband	White	Male	0	0	...	1	0	0	0	
2	38	215646	8	9	Handlers-cleaners	Not-in-family	White	Male	0	0	...	0	0	0	1	
3	53	234721	6	7	Handlers-cleaners	Husband	Black	Male	0	0	...	0	0	0	0	
4	28	338409	9	13	Prof-specialty	Wife	Black	Female	0	0	...	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
32556	27	257302	12	12	Tech-support	Wife	White	Female	0	0	...	0	0	0	0	
32557	40	154374	8	9	Machine-op-inspct	Husband	White	Male	0	0	...	0	0	0	0	
32558	58	151910	8	9	Adm-clerical	Unmarried	White	Female	0	0	...	0	0	0	0	
32559	22	201490	8	9	Adm-clerical	Own-child	White	Male	0	0	...	0	0	0	0	
32560	52	287927	8	9	Exec-managerial	Wife	White	Female	15024	0	...	0	0	0	0	

32561 rows × 28 columns

ENCODING DATA INTO CATEGORICAL TO NUMERICAL

```
dummies = pd.get_dummies(mer.occupation )
me = pd.concat([mer,dummies],axis=1)
me
```

	Age	Fnlwgt	Education	education_num	occupation	relationship	race	sex	capital_gain	capital_loss	...	Farming-fishing	Handlers-cleaners	Machine-op-inspct	Other-service
0	39	77516	9	13	Adm-clerical	Not-in-family	White	Male	2174	0	...	0	0	0	0
1	50	83311	9	13	Exec-managerial	Husband	White	Male	0	0	...	0	0	0	0
2	38	215646	8	9	Handlers-cleaners	Not-in-family	White	Male	0	0	...	0	1	0	0
3	53	234721	6	7	Handlers-cleaners	Husband	Black	Male	0	0	...	0	1	0	0
4	28	338409	9	13	Prof-specialty	Wife	Black	Female	0	0	...	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
32556	27	257302	12	12	Tech-support	Wife	White	Female	0	0	...	0	0	0	0
32557	40	154374	8	9	Machine-op-inspct	Husband	White	Male	0	0	...	0	0	1	0
32558	58	151910	8	9	Adm-clerical	Unmarried	White	Female	0	0	...	0	0	0	0
32559	22	201490	8	9	Adm-clerical	Own-child	White	Male	0	0	...	0	0	0	0
32560	52	287927	8	9	Exec-managerial	Wife	White	Female	15024	0	...	0	0	0	0

32561 rows × 42 columns

```
m = me.drop(['occupation'],axis =1)
m
```

	Age	Fnlwgt	Education	education_num	relationship	race	sex	capital_gain	capital_loss	hours_per_week	...	Farming-fishing	Handlers-cleaners	Machine-op-inspct	Ot ser
0	39	77516	9	13	Not-in-family	White	Male	2174	0	40	...	0	0	0	
1	50	83311	9	13	Husband	White	Male	0	0	13	...	0	0	0	
2	38	215646	8	9	Not-in-family	White	Male	0	0	40	...	0	1	0	
3	53	234721	6	7	Husband	Black	Male	0	0	40	...	0	1	0	
4	28	338409	9	13	Wife	Black	Female	0	0	40	...	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
32556	27	257302	12	12	Wife	White	Female	0	0	38	...	0	0	0	
32557	40	154374	8	9	Husband	White	Male	0	0	40	...	0	0	1	
32558	58	151910	8	9	Unmarried	White	Female	0	0	40	...	0	0	0	
32559	22	201490	8	9	Own-child	White	Male	0	0	20	...	0	0	0	
32560	52	287927	8	9	Wife	White	Female	15024	0	40	...	0	0	0	

32561 rows × 41 columns

ENCODING DATA INTO CATEGORICAL TO NUMERICAL

```
dummies = pd.get_dummies(mer.relationship)
remov = pd.concat([m,dummies],axis=1)
remov
```

	Age	Fnlwgt	Education	education_num	relationship	race	sex	capital_gain	capital_loss	hours_per_week	...	Protective-serv	Sales	Tech-support	Transp mov
0	39	77516	9	13	Not-in-family	White	Male	2174	0	40	...	0	0	0	
1	50	83311	9	13	Husband	White	Male	0	0	13	...	0	0	0	
2	38	215646	8	9	Not-in-family	White	Male	0	0	40	...	0	0	0	
3	53	234721	6	7	Husband	Black	Male	0	0	40	...	0	0	0	
4	28	338409	9	13	Wife	Black	Female	0	0	40	...	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
32556	27	257302	12	12	Wife	White	Female	0	0	38	...	0	0	1	
32557	40	154374	8	9	Husband	White	Male	0	0	40	...	0	0	0	
32558	58	151910	8	9	Unmarried	White	Female	0	0	40	...	0	0	0	
32559	22	201490	8	9	Own-child	White	Male	0	0	20	...	0	0	0	
32560	52	287927	8	9	Wife	White	Female	15024	0	40	...	0	0	0	

32561 rows × 47 columns

```
remove = remov.drop(['relationship'],axis =1)
remove
```

	Age	Fnlwgt	Education	education_num	race	sex	capital_gain	capital_loss	hours_per_week	native_country	...	Protective-serv	Sales	Tech-support	Tran n
0	39	77516	9	13	White	Male	2174	0	40	United-States	...	0	0	0	
1	50	83311	9	13	White	Male	0	0	13	United-States	...	0	0	0	
2	38	215646	8	9	White	Male	0	0	40	United-States	...	0	0	0	
3	53	234721	6	7	Black	Male	0	0	40	United-States	...	0	0	0	
4	28	338409	9	13	Black	Female	0	0	40	Cuba	...	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
32556	27	257302	12	12	White	Female	0	0	38	United-States	...	0	0	1	
32557	40	154374	8	9	White	Male	0	0	40	United-States	...	0	0	0	
32558	58	151910	8	9	White	Female	0	0	40	United-States	...	0	0	0	
32559	22	201490	8	9	White	Male	0	0	20	United-States	...	0	0	0	
32560	52	287927	8	9	White	Female	15024	0	40	United-States	...	0	0	0	

32561 rows × 46 columns

ENCODING DATA INTO CATEGORICAL TO NUMERICAL

```
dummies = pd.get_dummies(remove.native_country)
rem = pd.concat([remove,dummies],axis=1)
rem
```

	Age	Fnlwgt	Education	education_num	race	sex	capital_gain	capital_loss	hours_per_week	native_country	...	Portugal	Puerto-Rico	Scotland	Sou
0	39	77516	9	13	White	Male	2174	0	40	United-States	...	0	0	0	
1	50	83311	9	13	White	Male	0	0	13	United-States	...	0	0	0	
2	38	215646	8	9	White	Male	0	0	40	United-States	...	0	0	0	
3	53	234721	6	7	Black	Male	0	0	40	United-States	...	0	0	0	
4	28	338409	9	13	Black	Female	0	0	40	Cuba	...	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
32556	27	257302	12	12	White	Female	0	0	38	United-States	...	0	0	0	
32557	40	154374	8	9	White	Male	0	0	40	United-States	...	0	0	0	
32558	58	151910	8	9	White	Female	0	0	40	United-States	...	0	0	0	
32559	22	201490	8	9	White	Male	0	0	20	United-States	...	0	0	0	
32560	52	287927	8	9	White	Female	15024	0	40	United-States	...	0	0	0	

32561 rows × 87 columns

```
remo = rem.drop(['native_country'],axis =1)
remo
```

	Age	Fnlwgt	Education	education_num	race	sex	capital_gain	capital_loss	hours_per_week	income	...	Portugal	Puerto-Rico	Scotland	South	Tai
0	39	77516	9	13	White	Male	2174	0	40	0	...	0	0	0	0	
1	50	83311	9	13	White	Male	0	0	13	0	...	0	0	0	0	
2	38	215646	8	9	White	Male	0	0	40	0	...	0	0	0	0	
3	53	234721	6	7	Black	Male	0	0	40	0	...	0	0	0	0	
4	28	338409	9	13	Black	Female	0	0	40	0	...	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
32556	27	257302	12	12	White	Female	0	0	38	0	...	0	0	0	0	
32557	40	154374	8	9	White	Male	0	0	40	1	...	0	0	0	0	
32558	58	151910	8	9	White	Female	0	0	40	0	...	0	0	0	0	
32559	22	201490	8	9	White	Male	0	0	20	0	...	0	0	0	0	
32560	52	287927	8	9	White	Female	15024	0	40	1	...	0	0	0	0	

32561 rows × 86 columns

ENCODING DATA INTO CATEGORICAL TO NUMERICAL

```
dummies = pd.get_dummies(remo.race)
r = pd.concat([remo,dummies],axis=1)
r
```

	Age	Fnlwgt	Education	education_num	race	sex	capital_gain	capital_loss	hours_per_week	income	...	Thailand	Trinadad&Tobago	United-States	Vi
0	39	77516	9	13	White	Male	2174	0	40	0	...	0	0	1	
1	50	83311	9	13	White	Male	0	0	13	0	...	0	0	1	
2	38	215646	8	9	White	Male	0	0	40	0	...	0	0	1	
3	53	234721	6	7	Black	Male	0	0	40	0	...	0	0	1	
4	28	338409	9	13	Black	Female	0	0	40	0	...	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
32556	27	257302	12	12	White	Female	0	0	38	0	...	0	0	1	
32557	40	154374	8	9	White	Male	0	0	40	1	...	0	0	1	
32558	58	151910	8	9	White	Female	0	0	40	0	...	0	0	1	
32559	22	201490	8	9	White	Male	0	0	20	0	...	0	0	1	
32560	52	287927	8	9	White	Female	15024	0	40	1	...	0	0	1	

32561 rows × 91 columns

```
re = r.drop(['race'],axis =1)
re
```

	Age	Fnlwgt	Education	education_num	sex	capital_gain	capital_loss	hours_per_week	income	Federal-gov	...	Thailand	Trinadad&Tobago	United-States
0	39	77516	9	13	Male	2174	0	40	0	0	...	0	0	1
1	50	83311	9	13	Male	0	0	13	0	0	...	0	0	1
2	38	215646	8	9	Male	0	0	40	0	0	...	0	0	1
3	53	234721	6	7	Male	0	0	40	0	0	...	0	0	1
4	28	338409	9	13	Female	0	0	40	0	0	...	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
32556	27	257302	12	12	Female	0	0	38	0	0	...	0	0	1
32557	40	154374	8	9	Male	0	0	40	1	0	...	0	0	1
32558	58	151910	8	9	Female	0	0	40	0	0	...	0	0	1
32559	22	201490	8	9	Male	0	0	20	0	0	...	0	0	1
32560	52	287927	8	9	Female	15024	0	40	1	0	...	0	0	1

32561 rows × 90 columns

ENCODING DATA INTO CATEGORICAL TO NUMERICAL

```
dummies= pd.get_dummies(re.sex)
rom = pd.concat([re,dummies],axis=1)
rom
```

	Age	Fnlwgt	Education	education_num	sex	capital_gain	capital_loss	hours_per_week	income	Federal-gov	...	United-States	Vietnam	Yugoslavia	Amer Indian Eskim
0	39	77516	9	13	Male	2174	0	40	0	0	...	1	0	0	(
1	50	83311	9	13	Male	0	0	13	0	0	...	1	0	0	(
2	38	215646	8	9	Male	0	0	40	0	0	...	1	0	0	(
3	53	234721	6	7	Male	0	0	40	0	0	...	1	0	0	(
4	28	338409	9	13	Female	0	0	40	0	0	...	0	0	0	(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
32556	27	257302	12	12	Female	0	0	38	0	0	...	1	0	0	(
32557	40	154374	8	9	Male	0	0	40	1	0	...	1	0	0	(
32558	58	151910	8	9	Female	0	0	40	0	0	...	1	0	0	(
32559	22	201490	8	9	Male	0	0	20	0	0	...	1	0	0	(
32560	52	287927	8	9	Female	15024	0	40	1	0	...	1	0	0	(

32561 rows × 92 columns

```
data= rom.drop(['sex',' Female'],axis =1)
data
```

	Age	Fnlwgt	Education	education_num	capital_gain	capital_loss	hours_per_week	income	Federal-gov	Local-gov	...	Trinadad&Tobago	United-States	Vietnam	Y
0	39	77516	9	13	2174	0	40	0	0	0	...	0	1	0	
1	50	83311	9	13	0	0	13	0	0	0	...	0	1	0	
2	38	215646	8	9	0	0	40	0	0	0	...	0	1	0	
3	53	234721	6	7	0	0	40	0	0	0	...	0	1	0	
4	28	338409	9	13	0	0	40	0	0	0	...	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
32556	27	257302	12	12	0	0	38	0	0	0	...	0	1	0	
32557	40	154374	8	9	0	0	40	1	0	0	...	0	1	0	
32558	58	151910	8	9	0	0	40	0	0	0	...	0	1	0	
32559	22	201490	8	9	0	0	20	0	0	0	...	0	1	0	
32560	52	287927	8	9	15024	0	40	1	0	0	...	0	1	0	

32561 rows × 90 columns

CORRELATION MATRIX

```
corr_mat = data.corr()
corr_mat
```

	Age	Fnlwgt	Education	education_num	capital_gain	capital_loss	hours_per_week	income	Federal-gov	Local-gov	...	Trinadad&Tobag
Age	1.000000	-0.076646	-0.031481	0.036527	0.077674	0.057775	0.068756	0.234037	0.051227	0.060901	...	0.0048
Fnlwgt	-0.076646	1.000000	-0.040967	-0.043195	0.000432	-0.010252	-0.018768	-0.009463	-0.007525	-0.002828	...	0.0052
Education	-0.031481	-0.040967	1.000000	0.741512	0.087865	0.046800	0.084405	0.203871	0.054994	0.038489	...	-0.0143
education_num	0.036527	-0.043195	0.741512	1.000000	0.122630	0.079923	0.148123	0.335154	0.060518	0.097941	...	-0.0170
capital_gain	0.077674	0.000432	0.087865	0.122630	1.000000	-0.031615	0.078409	0.223329	-0.005768	-0.007007	...	-0.0035
...	...	...	...	...	...	...	...	...	...	...	...	
Asian-Pac-Islander	-0.011111	-0.051323	0.028007	0.062091	0.009851	0.004469	-0.004564	0.010543	0.013808	-0.019797	...	0.0100
Black	-0.019434	0.118009	-0.044060	-0.075272	-0.020631	-0.021762	-0.053153	-0.089089	0.047403	0.037073	...	0.0612
Other	-0.034415	0.006376	-0.043145	-0.044133	-0.001774	-0.005964	-0.007188	-0.031830	-0.001978	-0.010227	...	0.0117
White	0.033412	-0.056896	0.036857	0.051353	0.014429	0.021044	0.049345	0.085224	-0.050995	-0.024132	...	-0.0585
Male	0.088832	0.026858	-0.026747	0.012280	0.048480	0.045567	0.229309	0.215980	0.000989	-0.037966	...	-0.0127

90 rows × 90 columns

IMPORTING REQUIRED MODULES FOR MODELS

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from sklearn.metrics import mean_squared_error
```

3) Split data into training and test data.

```
#SPLITTING THE DATA

# get all the features
features = [feat for feat in data.columns if feat !='income']

X = data[features] # feature set
Y = data['income'] # target

# Splitting data into train and test
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.20, random_state=1, stratify=Y)

# train and test datasets dimensions
X_train.shape, X_test.shape
```

```
((26048, 89), (6513, 89))
```

```
num_features=data.columns.drop('income')
num_features
```

```
Index(['Age', 'Fnlwgt', 'Education', 'education_num', 'capital_gain',
       'capital_loss', 'hours_per_week', ' Federal-gov', ' Local-gov',
       ' Never-worked', ' Private', ' Self-emp-inc', ' Self-emp-not-inc',
       ' State-gov', ' Without-pay', ' Divorced', ' Married-AF-spouse',
       ' Married-civ-spouse', ' Married-spouse-absent', ' Never-married',
       ' Separated', ' Widowed', ' Adm-clerical', ' Armed-Forces',
       ' Craft-repair', ' Exec-managerial', ' Farming-fishing',
       ' Handlers-cleaners', ' Machine-op-inspct', ' Other-service',
       ' Priv-house-serv', ' Prof-specialty', ' Protective-serv', ' Sales',
       ' Tech-support', ' Transport-moving', ' Husband', ' Not-in-family',
       ' Other-relative', ' Own-child', ' Unmarried', ' Wife', ' Cambodia',
       ' Canada', ' China', ' Columbia', ' Cuba', ' Dominican-Republic',
       ' Ecuador', ' El-Salvador', ' England', ' France', ' Germany',
       ' Greece', ' Guatemala', ' Haiti', ' Holand-Netherlands', ' Honduras',
       ' Hong', ' Hungary', ' India', ' Iran', ' Ireland', ' Italy',
       ' Jamaica', ' Japan', ' Laos', ' Mexico', ' Nicaragua',
       ' Outlying-US(Guam-USVI-etc)', ' Peru', ' Philippines', ' Poland',
       ' Portugal', ' Puerto-Rico', ' Scotland', ' South', ' Taiwan',
       ' Thailand', ' Trinadad&Tobago', ' United-States', ' Vietnam',
       ' Yugoslavia', ' Amer-Indian-Eskimo', ' Asian-Pac-Islander', ' Black',
       ' Other', ' White', ' Male'],
      dtype='object')
```

SCALING THE DATA

```
# MinMaxScaler will scale the features to a range of [0, 1]

scaler = MinMaxScaler(feature_range=(0, 1))

X_train[num_features] = scaler.fit_transform(X_train[num_features]) #fit and transform the train set
X_test[num_features] = scaler.transform(X_test[num_features]) #transform the test set
```

```
C:\Users\DHAVEER\anaconda3\lib\site-packages\pandas\core\frame.py:3678: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy
  self[col] = igetitem(value, i)
C:\Users\DHAVEER\anaconda3\lib\site-packages\pandas\core\frame.py:3678: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy
  self[col] = igetitem(value, i)
```

```
sc = MinMaxScaler()
df1 = sc.fit_transform(data)
df1
```

```
array([[0.30136986, 0.0443019 , 0.6       , ..., 0.        , 1.        ,
        1.        ],
       [0.45205479, 0.0482376 , 0.6       , ..., 0.        , 1.        ,
        1.        ],
       [0.28767123, 0.13811345, 0.53333333, ..., 0.        , 1.        ,
        1.        ],
       ...,
       [0.56164384, 0.09482688, 0.53333333, ..., 0.        , 1.        ,
        0.        ],
       [0.06849315, 0.12849934, 0.53333333, ..., 0.        , 1.        ,
        1.        ],
       [0.47945205, 0.18720338, 0.53333333, ..., 0.        , 1.        ,
        0.        ]])
```

```
df2=pd.DataFrame(df1)
df2
```

	0	1	2	3	4	5	6	7	8	9	...	80	81	82	83	84	85	86	87	88	89
0	0.301370	0.044302	0.600000	0.800000	0.021740	0.0	0.397959	0.0	0.0	0.0	...	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
1	0.452055	0.048238	0.600000	0.800000	0.000000	0.0	0.122449	0.0	0.0	0.0	...	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
2	0.287671	0.138113	0.533333	0.533333	0.000000	0.0	0.397959	0.0	0.0	0.0	...	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
3	0.493151	0.151068	0.400000	0.400000	0.000000	0.0	0.397959	0.0	0.0	0.0	...	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
4	0.150685	0.221488	0.600000	0.800000	0.000000	0.0	0.397959	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
32556	0.136986	0.166404	0.800000	0.733333	0.000000	0.0	0.377551	0.0	0.0	0.0	...	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
32557	0.315068	0.096500	0.533333	0.533333	0.000000	0.0	0.397959	1.0	0.0	0.0	...	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
32558	0.561644	0.094827	0.533333	0.533333	0.000000	0.0	0.397959	0.0	0.0	0.0	...	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
32559	0.068493	0.128499	0.533333	0.533333	0.000000	0.0	0.193878	0.0	0.0	0.0	...	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
32560	0.479452	0.187203	0.533333	0.533333	0.150242	0.0	0.397959	1.0	0.0	0.0	...	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0

32561 rows × 90 columns

CONFUSION MATRIX, CLASSIFICATION REPORT, ACCURACY SCORE & MISCLASSIFICATION RATE

```
def model_perf(model,X_train,X_test,Y_train,Y_test):
    model.fit(X_train,Y_train)
    Y_pred =model.predict(X_test)
    print(Y_pred)

    #TRAINING & TESTING SCORE
    print("Training score : ", model.score(X_train,Y_train))
    print("Test score : ", model.score(X_test,Y_test))

    #CONFUSION MATRIX & CLASSIFICATION REPORT
    cm = confusion_matrix(Y_test,Y_pred)
    print('Confusion Matrix\n',cm)
    print('Classification report\n',classification_report(Y_test,Y_pred))

    #ACCURACY SCORE & MISCLASSIFICATION RATE
    print('Accuracy Score:', accuracy_score(Y_test,Y_pred))
    m = (1 - accuracy_score(Y_test,Y_pred))*100
    print("Misclassification rate ", round(m,2) , "%")
```

4) Apply the following models on the training dataset and generate the predicted value for the test dataset
5) Predict the income for test data
6) Compute Confusion matrix and classification report for each of these models.
7) Validate the result for Precision, Recall, F1-score and Accuracy for each model based on values from confusion_matrix and classification_report
8) Generate the percentage of misclassification in each of these models.

a. Decision Tree

```
#DECISION TREE

dtree = DecisionTreeClassifier(random_state=1, criterion='entropy', max_depth = 14, min_samples_split = 30)
dtree.fit(X_train,Y_train)
```

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=14,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=30,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=1, splitter='best')
```

```
model_perf(dtree,X_train,X_test,Y_train,Y_test)
```

```
[0 0 1 ... 1 0 1]
Training score :  0.8773418304668305
Test score :  0.8455396898510671
Confusion Matrix
 [[4585  360]
 [ 646  922]]
Classification report
              precision    recall  f1-score   support

           0       0.88      0.93      0.90      4945
           1       0.72      0.59      0.65      1568

    accuracy                           0.85      6513
   macro avg       0.80      0.76      0.77      6513
weighted avg       0.84      0.85      0.84      6513

Accuracy Score: 0.8455396898510671
Misclassification rate  15.45 %
```

b. Random Forest Classifier

```
#RANDOM FOREST

forest = RandomForestClassifier(random_state=1, n_estimators=150,min_samples_split=10,max_depth=10)
forest.fit(X_train,Y_train)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=10, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=10,
                       min_weight_fraction_leaf=0.0, n_estimators=150,
                       n_jobs=None, oob_score=False, random_state=1, verbose=0,
                       warm_start=False)
```

```
model_perf(forest,X_train,X_test,Y_train,Y_test)
```

```
[0 0 1 ... 1 0 1]
Training score :  0.8646729115479116
Test score :  0.8539843390142792
Confusion Matrix
 [[4711  234]
 [ 717  851]]
Classification report
               precision    recall  f1-score   support

           0       0.87      0.95      0.91      4945
           1       0.78      0.54      0.64      1568

    accuracy                           0.85      6513
   macro avg       0.83      0.75      0.77      6513
weighted avg       0.85      0.85      0.84      6513

Accuracy Score: 0.8539843390142792
Misclassification rate  14.6 %
```

c. Logistic Regression

```
#LOGISTIC REGRESSION

log_reg = LogisticRegression(max_iter=1000, solver='liblinear')
log_reg.fit(X_train,Y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=1000,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
model_perf(log_reg,X_train,X_test,Y_train,Y_test)
```

```
[0 0 1 ... 1 0 1]
Training score :  0.8512361793611793
Test score :  0.8479963150621833
Confusion Matrix
 [[4596  349]
 [ 641  927]]
Classification report
               precision    recall  f1-score   support

           0       0.88      0.93      0.90      4945
           1       0.73      0.59      0.65      1568

    accuracy                           0.85      6513
   macro avg       0.80      0.76      0.78      6513
weighted avg       0.84      0.85      0.84      6513

Accuracy Score: 0.8479963150621833
Misclassification rate  15.2 %
```

d. KNN Classifier

```
#K-NEAREST NEIGHBORS

kNN = KNeighborsClassifier(n_neighbors = 3)
kNN.fit(X_train,Y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
```

```
model_perf(kNN,X_train,X_test,Y_train,Y_test)
```

```
[0 0 0 ... 0 0 1]
Training score :  0.895078316953317
Test score :  0.8212805158912944
Confusion Matrix
 [[4452  493]
 [ 671  897]]
Classification report
               precision    recall  f1-score   support

           0       0.87      0.90      0.88      4945
           1       0.65      0.57      0.61      1568

    accuracy                           0.82      6513
   macro avg       0.76      0.74      0.75      6513
weighted avg       0.82      0.82      0.82      6513

Accuracy Score: 0.8212805158912944
Misclassification rate  17.87 %
```

e. SVC Classifier (with linear kernel)

```
#SUPPORT VECTOR MACHINE

svm = SVC(kernel='linear', C=100,  gamma = 0.1, degree = 4)
svm.fit(X_train,Y_train)
```

```
SVC(C=100, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=4, gamma=0.1, kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
model_perf(svm,X_train,X_test,Y_train,Y_test)
```

```
[0 0 1 ... 1 0 1]
Training score :  0.8511977886977887
Test score :  0.844311377245509
Confusion Matrix
 [[4602  343]
 [ 671  897]]
Classification report
              precision    recall  f1-score   support

           0       0.87      0.93      0.90      4945
           1       0.72      0.57      0.64      1568

    accuracy                           0.84      6513
   macro avg       0.80      0.75      0.77      6513
weighted avg       0.84      0.84      0.84      6513

Accuracy Score: 0.844311377245509
Misclassification rate  15.57 %
```

CONCLUSION:

The accuracy scores has been calculated for the adult income data using different models like Decision Tree, Random Forest, k-Nearest Neighbour, Logistic Regression, Support Vector Machine.
And found that the accuracy score is best for Random Forest(85.39%).
Hence, we conclude that RANDOM FOREST is the best fit model.