Census Income

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

```
In [2]: data = pd.read_csv("adult.data.csv")
        data.head()
```

	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
2	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
3	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
4	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States	<=50K

```
In [3]: data.columns  = ["age" ,"workclass" ,"fnlwgt" , "education" , "education_num" , "marital_status" , "occupation" , "relationship" , "race" ,
                    "sex" ,"capital_gain" , "capital_loss" , "hours_per_week" , "native_country" , "income"]
```

```
In [4]: data.head()
```

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	capital_gain
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0
2	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0
3	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0
4	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0

Data Information

```
In [5]: data.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
```

```
In [6]: data.workclass.nunique()

9
```

```
In [7]: data.workclass.value_counts()

 Private             22696
 Self-emp-not-inc     2541
 Local-gov            2093
 ?                    1836
 State-gov            1297
 Self-emp-inc         1116
 Federal-gov           960
 Without-pay            14
 Never-worked            7
Name: workclass, dtype: int64
```

```
In [8]: data.education.nunique()

16
```

```
In [9]: data.education.value_counts()

 HS-grad         10501
 Some-college     7291
 Bachelors        5354
 Masters          1723
 Assoc-voc        1382
 11th             1175
 Assoc-acdm       1067
 10th              933
 7th-8th           646
 Prof-school       576
 9th               514
 12th              433
 Doctorate         413
 5th-6th           333
 1st-4th           168
 Preschool          51
Name: education, dtype: int64
```

```
In [10]: data.occupation.value_counts()

 Prof-specialty       4140
 Craft-repair         4099
 Exec-managerial      4066
 Adm-clerical         3769
 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
```

```
In [11]: data.occupation.nunique()

15
```

```
In [12]: data.relationship.nunique()

6
```

```
In [13]: data.relationship.value_counts()

 Husband           13193
 Not-in-family      8304
 Own-child          5068
 Unmarried          3446
 Wife               1568
 Other-relative      981
Name: relationship, dtype: int64
```

```
In [14]: data.race.nunique()

5
```

```
In [15]: data.race.value_counts()

 White                 27815
 Black                  3124
 Asian-Pac-Islander     1039
 Amer-Indian-Eskimo      311
 Other                   271
Name: race, dtype: int64
```

```
In [16]: data.sex.value_counts()

 Male      21789
 Female    10771
Name: sex, dtype: int64
```

```
In [17]: data.native_country.value_counts()

 United-States                 29169
 Mexico                          643
 ?                               583
 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
 Thailand                         18
 Laos                             18
 Yugoslavia                       16
 Outlying-US(Guam-USVI-etc)       14
 Hungary                          13
 Honduras                         13
 Scotland                         12
 Holand-Netherlands                1
Name: native_country, dtype: int64
```

```
In [18]: data.income.value_counts()

 <=50K    24719
 >50K      7841
Name: income, dtype: int64
```

```
In [19]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32560 entries, 0 to 32559
Data columns (total 15 columns):
age               32560 non-null int64
workclass         32560 non-null object
fnlwgt            32560 non-null int64
education         32560 non-null object
education_num     32560 non-null int64
marital_status    32560 non-null object
occupation        32560 non-null object
relationship      32560 non-null object
race              32560 non-null object
sex               32560 non-null object
capital_gain      32560 non-null int64
capital_loss      32560 non-null int64
hours_per_week    32560 non-null int64
native_country    32560 non-null object
income            32560 non-null object
dtypes: int64(6), object(9)
memory usage: 2.6+ MB
```

```
In [20]: data.describe()
```

	age	fnlwgt	education_num	capital_gain	capital_loss	hours_per_week
count	32560.000000	3.256000e+04	32560.000000	32560.000000	32560.000000	32560.000000
mean	38.581634	1.897818e+05	10.080590	1077.615172	87.306511	40.437469
std	13.640642	1.055498e+05	2.572709	7385.402999	402.966116	12.347618
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178315e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783630e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370545e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

Processing Columns with ?

```
In [21]: data["workclass"]  =  data["workclass"].replace( '[\?,)]','workclass_Not_available', regex=True )
         data["occupation"]  =  data["occupation"].replace( '[\?,)]','occupation_Not_available', regex=True )
         data["native_country"]  =  data["native_country"].replace( '[\?,)]','native_country_Not_available', regex=True )
```

Handling Data with Cardinality above 15

```
In [22]: data.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)
```

```
In [23]: data["education"] = data["education"].replace([' 1st-4th' , ' 5th-6th',' Preschool' , ' 7th-8th' ], "Below 9th class")
         data["education"] = data["education"].replace([' 9th',' 10th', ' 11th',' 12th' ], "9th and Above")
```

```
In [24]: data.education.unique()

array([' Bachelors', ' HS-grad', '9th and Above', ' Masters',
       ' Some-college', ' Assoc-acdm', ' Assoc-voc', 'Below 9th class',
       ' Doctorate', ' Prof-school'], dtype=object)
```

```
In [25]: data.education.nunique()

10
```

Handling Object and Numerical Columns Seperately

```
In [26]: s = (data.dtypes == "object")
         obj_cols =list(s[s].index)
         obj_cols

['workclass',
 'education',
 'marital_status',
 'occupation',
 'relationship',
 'race',
 'sex',
 'native_country',
 'income']
```

```
In [27]: data.head()
```

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	capital_gain
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0
2	53	Private	234721	9th and Above	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0
3	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0
4	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0

```
In [28]: data_with_num = data.select_dtypes(exclude = ["object"])
         data_with_num.head()
```

	age	fnlwgt	education_num	capital_gain	capital_loss	hours_per_week
0	50	83311	13	0	0	13
1	38	215646	9	0	0	40
2	53	234721	7	0	0	40
3	28	338409	13	0	0	40
4	37	284582	14	0	0	40

Handling Categorical Value

```
In [29]: data_with_obj = data[obj_cols]
         data_with_obj.head()
```

	workclass	education	marital_status	occupation	relationship	race	sex	native_country	income
0	Self-emp-not-inc	Bachelors	Married-civ-spouse	Exec-managerial	Husband	White	Male	United-States	<=50K
1	Private	HS-grad	Divorced	Handlers-cleaners	Not-in-family	White	Male	United-States	<=50K
2	Private	9th and Above	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	United-States	<=50K
3	Private	Bachelors	Married-civ-spouse	Prof-specialty	Wife	Black	Female	Cuba	<=50K
4	Private	Masters	Married-civ-spouse	Exec-managerial	Wife	White	Female	United-States	<=50K

```
In [30]: print(f'Workclass {data["workclass"].nunique()} \n Education: {data["education"].nunique()} \n Martial Status {data.marital_status.nunique()} \n Occupation {data.occupation.nunique()} \n Relationship {data.relationship.nunique()} \n Race {data.race.nunique()} \n Sex {data.sex.nunique()} \n Country {data.native_country.nunique()} \n Income {data.income.nunique()}')

Workclass 9
 Education: 10
 Martial Status 7
 Occupation 15
 Relationship 6
 Race 5
 Sex 2
 Country 42
 Income 2
```

```
In [31]: data_with_obj.columns

Index(['workclass', 'education', 'marital_status', 'occupation',
       'relationship', 'race', 'sex', 'native_country', 'income'],
      dtype='object')
```

```
In [32]: two_class = ["sex" ,"income"]
         multiple_classes = ['workclass', 'education', 'marital_status', 'occupation',
                'relationship', 'race']
```

```
In [33]: from sklearn.preprocessing import LabelEncoder
```

```
In [34]: data_two_class = pd.DataFrame({
             "sex" :  LabelEncoder().fit_transform(data_with_obj["sex"]),
             "income" :LabelEncoder().fit_transform(data_with_obj["income"]),
         })
```

```
In [35]: data_two_class.head()
```

	sex	income
0	1	0
1	1	0
2	1	0
3	0	0
4	0	0

```
In [36]: data_multiple_classes = pd.get_dummies(data_with_obj[multiple_classes])
```

```
In [37]: data_multiple_classes.head()
```

	workclass_ Federal-gov	workclass_ Local-gov	workclass_ Never-worked	workclass_ Private	workclass_ Self-emp-inc	workclass_ Self-emp-not-inc	workclass_ State-gov	workclass_ Without-pay	workclass_ workclass_Not_avai
0	0	0	0	0	0	1	0	0	0
1	0	0	0	1	0	0	0	0	0
2	0	0	0	1	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0

5 rows × 52 columns

```
In [38]: data_multiple_classes.columns

Index(['workclass_ Federal-gov', 'workclass_ Local-gov',
       'workclass_ Never-worked', 'workclass_ Private',
       'workclass_ Self-emp-inc', 'workclass_ Self-emp-not-inc',
       'workclass_ State-gov', 'workclass_ Without-pay',
       'workclass_ workclass_Not_available', 'education_ Assoc-acdm',
       'education_ Assoc-voc', 'education_ Bachelors', 'education_ Doctorate',
       'education_ HS-grad', 'education_ Masters', 'education_ Prof-school',
       'education_ Some-college', 'education_9th and Above',
       'education_Below 9th class', 'marital_status_ Divorced',
       'marital_status_ Married-AF-spouse',
       'marital_status_ Married-civ-spouse',
       'marital_status_ Married-spouse-absent',
       'marital_status_ Never-married', 'marital_status_ Separated',
       'marital_status_ Widowed', 'occupation_ Adm-clerical',
       'occupation_ Armed-Forces', 'occupation_ Craft-repair',
       'occupation_ Exec-managerial', 'occupation_ Farming-fishing',
       'occupation_ Handlers-cleaners', 'occupation_ Machine-op-inspct',
       'occupation_ Other-service', 'occupation_ Priv-house-serv',
       'occupation_ Prof-specialty', 'occupation_ Protective-serv',
       'occupation_ Sales', 'occupation_ Tech-support',
       'occupation_ Transport-moving', 'occupation_ occupation_Not_available',
       'relationship_ Husband', 'relationship_ Not-in-family',
       'relationship_ Other-relative', 'relationship_ Own-child',
       'relationship_ Unmarried', 'relationship_ Wife',
       'race_ Amer-Indian-Eskimo', 'race_ Asian-Pac-Islander', 'race_ Black',
       'race_ Other', 'race_ White'],
      dtype='object')
```

```
In [39]: lst = [data_two_class , data_multiple_classes]
         final_obj_data = pd.concat(lst , axis = 1)
         final_obj_data.head()
```

	sex	income	workclass_ Federal-gov	workclass_ Local-gov	workclass_ Never-worked	workclass_ Private	workclass_ Self-emp-inc	workclass_ Self-emp-not-inc	workclass_ State-gov	workclass_ Without-pay	...
0	1	0	0	0	0	0	0	1	0	0	...
1	1	0	0	0	0	1	0	0	0	0	...
2	1	0	0	0	0	1	0	0	0	0	...
3	0	0	0	0	0	1	0	0	0	0	...
4	0	0	0	0	0	1	0	0	0	0	...

5 rows × 54 columns

Handling Numerical Values

```
In [40]: data_with_num.head()
```

	age	fnlwgt	education_num	capital_gain	capital_loss	hours_per_week
0	50	83311	13	0	0	13
1	38	215646	9	0	0	40
2	53	234721	7	0	0	40
3	28	338409	13	0	0	40
4	37	284582	14	0	0	40

```
In [41]: data_with_num["hours_per_week"].unique()

array([13, 40, 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)
```

```
In [42]: data_with_num["age"].unique()

array([50, 38, 53, 28, 37, 49, 52, 31, 42, 30, 23, 32, 40, 34, 25, 43, 54,
       35, 59, 56, 19, 39, 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)
```

```
In [43]: data_with_num["education_num"].unique()

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

```
In [44]: data_with_num.hours_per_week = data.hours_per_week/100
         data_with_num.age = data_with_num.age/100
         data_with_num.education_num = data_with_num.education_num/100

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py:5096: 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: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  self[name] = value
```

```
In [45]: data_with_num.head()
```

	age	fnlwgt	education_num	capital_gain	capital_loss	hours_per_week
0	0.50	83311	0.13	0	0	0.13
1	0.38	215646	0.09	0	0	0.40
2	0.53	234721	0.07	0	0	0.40
3	0.28	338409	0.13	0	0	0.40
4	0.37	284582	0.14	0	0	0.40

```
In [46]: from sklearn.preprocessing import MinMaxScaler
         sc = MinMaxScaler()
```

```
In [47]: lst_scaler_values = ["fnlwgt" ,"capital_gain" , "capital_loss"]
         data_array = sc.fit_transform(data_with_num[lst_scaler_values])
         data_array[:5]

array([[0.0482376 , 0.        , 0.        ],
       [0.13811345, 0.        , 0.        ],
       [0.15106831, 0.        , 0.        ],
       [0.22148843, 0.        , 0.        ],
       [0.18493161, 0.        , 0.        ]])
```

```
In [48]: column_values = ["fnlwgt","capital_gain","capital_loss"]
         df = pd.DataFrame(data = data_array,
                           columns = column_values)
         df.tail()
```

	fnlwgt	capital_gain	capital_loss
32555	0.166404	0.000000	0.0
32556	0.096500	0.000000	0.0
32557	0.094827	0.000000	0.0
32558	0.128499	0.000000	0.0
32559	0.187203	0.150242	0.0

```
In [49]: lst_num_ =[data_with_num[["age","hours_per_week" , "education_num"]] , df]
         final_num_data = pd.concat(lst_num_ , axis = 1)
```

```
In [50]: final_num_data.head()
```

	age	hours_per_week	education_num	fnlwgt	capital_gain	capital_loss
0	0.50	0.13	0.13	0.048238	0.0	0.0
1	0.38	0.40	0.09	0.138113	0.0	0.0
2	0.53	0.40	0.07	0.151068	0.0	0.0
3	0.28	0.40	0.13	0.221488	0.0	0.0
4	0.37	0.40	0.14	0.184932	0.0	0.0

```
In [51]: final_df_lst = [final_num_data , final_obj_data]
         final_data = pd.concat(final_df_lst , axis = 1)
         final_data.head()
```

	age	hours_per_week	education_num	fnlwgt	capital_gain	capital_loss	sex	income	workclass_ Federal-gov	workclass_ Local-gov	...
0	0.50	0.13	0.13	0.048238	0.0	0.0	1	0	0	0	...
1	0.38	0.40	0.09	0.138113	0.0	0.0	1	0	0	0	...
2	0.53	0.40	0.07	0.151068	0.0	0.0	1	0	0	0	...
3	0.28	0.40	0.13	0.221488	0.0	0.0	0	0	0	0	...
4	0.37	0.40	0.14	0.184932	0.0	0.0	0	0	0	0	...

5 rows × 60 columns

```
In [52]: final_data.columns

Index(['age', 'hours_per_week', 'education_num', 'fnlwgt', 'capital_gain',
       'capital_loss', 'sex', 'income', 'workclass_ Federal-gov',
       'workclass_ Local-gov', 'workclass_ Never-worked', 'workclass_ Private',
       'workclass_ Self-emp-inc', 'workclass_ Self-emp-not-inc',
       'workclass_ State-gov', 'workclass_ Without-pay',
       'workclass_ workclass_Not_available', 'education_ Assoc-acdm',
       'education_ Assoc-voc', 'education_ Bachelors', 'education_ Doctorate',
       'education_ HS-grad', 'education_ Masters', 'education_ Prof-school',
       'education_ Some-college', 'education_9th and Above',
       'education_Below 9th class', 'marital_status_ Divorced',
       'marital_status_ Married-AF-spouse',
       'marital_status_ Married-civ-spouse',
       'marital_status_ Married-spouse-absent',
       'marital_status_ Never-married', 'marital_status_ Separated',
       'marital_status_ Widowed', 'occupation_ Adm-clerical',
       'occupation_ Armed-Forces', 'occupation_ Craft-repair',
       'occupation_ Exec-managerial', 'occupation_ Farming-fishing',
       'occupation_ Handlers-cleaners', 'occupation_ Machine-op-inspct',
       'occupation_ Other-service', 'occupation_ Priv-house-serv',
       'occupation_ Prof-specialty', 'occupation_ Protective-serv',
       'occupation_ Sales', 'occupation_ Tech-support',
       'occupation_ Transport-moving', 'occupation_ occupation_Not_available',
       'relationship_ Husband', 'relationship_ Not-in-family',
       'relationship_ Other-relative', 'relationship_ Own-child',
       'relationship_ Unmarried', 'relationship_ Wife',
       'race_ Amer-Indian-Eskimo', 'race_ Asian-Pac-Islander', 'race_ Black',
       'race_ Other', 'race_ White'],
      dtype='object')
```

Training Model

```
In [53]: from sklearn import svm
         from sklearn.ensemble import RandomForestClassifier
         from sklearn.tree import DecisionTreeClassifier
         from sklearn.linear_model import LogisticRegression

         model_params = {
             'svm': {
                 'model': svm.SVC(gamma='auto'),
                 'params' : {
                     'C': [1,10,20],
                     'kernel': ['rbf','linear']
                 }
             },
             'random_forest': {
                 'model': RandomForestClassifier(),
                 'params' : {
                     'n_estimators': [1,5,10]
                 }
             },
             'logistic_regression' : {
                 'model': LogisticRegression(solver='liblinear',multi_class='auto'),
                 'params': {
                     'C': [1,5,10]
                 }
             },
             'decision_tree':{
                 'model': DecisionTreeClassifier(),
                 'params':{
                     'criterion' : ["gini", "entropy"],
                     'max_depth' :[10 , 50 , 100 , 500 , 1000]
                 }
             }
         }
```

```
In [54]: X = final_data.drop("income" ,axis =1)
         Y = final_data.income
```

Lowering The Dimension

```
In [55]: from sklearn.decomposition import PCA
         pca = PCA(n_components= 3)
         transformed_data = pca.fit_transform(X)
         transformed_data

array([[-1.15548434,  0.81813587,  0.21231776],
       [ 0.49630389, -0.77958991,  0.5438017 ],
       [-0.8077047 , -0.18663856, -0.25071512],
       ...,
       [ 0.62166239, -0.79104592,  0.83892539],
       [ 0.75181831, -0.79227277, -0.68583418],
       [-0.2764336 , -0.09509612,  0.94402833]])
```

Training Different Model and Selecting the Best Model

```
In [56]: from sklearn.model_selection import GridSearchCV
         scores = []

         for model_name, mp in model_params.items():
             clf =  GridSearchCV(mp['model'], mp['params'], cv=5, return_train_score=False)
             clf.fit(transformed_data,Y)
             scores.append({
                 'model': model_name,
                 'best_score': clf.best_score_,
                 'best_params': clf.best_params_
             })

         df_grid = pd.DataFrame(scores,columns=['model','best_score','best_params'])
         df_grid
```

	model	best_score	best_params
0	svm	0.805436	{'C': 20, 'kernel': 'rbf'}
1	random_forest	0.810104	{'n_estimators': 10}
2	logistic_regression	0.777733	{'C': 1}
3	decision_tree	0.815080	{'criterion': 'entropy', 'max_depth': 10}

Handling Test Data

```
In [57]: test_data = pd.read_csv("adult.test.csv")
         test_data.head()
```

	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0.1	40	United-States	<=50K.
0	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States	<=50K.
1	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States	>50K.
2	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	United-States	>50K.
3	18	?	103497	Some-college	10	Never-married	?	Own-child	White	Female	0	0	30	United-States	<=50K.
4	34	Private	198693	10th	6	Never-married	Other-service	Not-in-family	White	Male	0	0	30	United-States	<=50K.

```
In [58]: test_data.columns  = ["age" ,"workclass" ,"fnlwgt" , "education" , "education_num" , "marital_status" , "occupation" , "relationship" , "race" ,
                    "sex" ,"capital_gain" , "capital_loss" , "hours_per_week" , "native_country" , "income"]
```

```
In [59]: test_data.head()
```

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	capital_gain
0	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0
1	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0
2	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688
3	18	?	103497	Some-college	10	Never-married	?	Own-child	White	Female	0
4	34	Private	198693	10th	6	Never-married	Other-service	Not-in-family	White	Male	0

Processing Test Data

```
In [60]: test_data["workclass"]  =  test_data["workclass"].replace( '[\?,)]','workclass_Not_available', regex=True )
         test_data["occupation"]  =  test_data["occupation"].replace( '[\?,)]','occupation_Not_available', regex=True )
         test_data["native_country"]  = test_data["native_country"].replace( '[\?,)]','native_country_Not_available', regex=True )
```

```
In [61]: test_data["education"] = test_data["education"].replace([' 1st-4th' , ' 5th-6th',' Preschool' , ' 7th-8th' ], "Below 9th class")
         test_data["education"] = test_data["education"].replace([' 9th',' 10th', ' 11th',' 12th' ], "9th and Above")
```

```
In [62]: test_s = (test_data.dtypes == "object")
         test_obj_cols =list(test_s[test_s].index)
         test_obj_cols

['workclass',
 'education',
 'marital_status',
 'occupation',
 'relationship',
 'race',
 'sex',
 'native_country',
 'income']
```

```
In [63]: test_data_with_num = test_data.select_dtypes(exclude = ["object"])
         test_data_with_num.head()
```

	age	fnlwgt	education_num	capital_gain	capital_loss	hours_per_week
0	38	89814	9	0	0	50
1	28	336951	12	0	0	40
2	44	160323	10	7688	0	40
3	18	103497	10	0	0	30
4	34	198693	6	0	0	30

```
In [64]: test_data_with_obj = test_data[test_obj_cols]
         test_data_with_obj.head()
```

	workclass	education	marital_status	occupation	relationship	race	sex	native_country	incom
0	Private	HS-grad	Married-civ-spouse	Farming-fishing	Husband	White	Male	United-States	<=50K.
1	Local-gov	Assoc-acdm	Married-civ-spouse	Protective-serv	Husband	White	Male	United-States	>50K.
2	Private	Some-college	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	United-States	>50K.
3	workclass_Not_available	Some-college	Never-married	occupation_Not_available	Own-child	White	Female	United-States	<=50K.
4	Private	9th and Above	Never-married	Other-service	Not-in-family	White	Male	United-States	<=50K.

```
In [65]: test_two_class = ["sex" ,"income"]
         test_multiple_classes = ['workclass', 'education', 'marital_status', 'occupation',
                'relationship', 'race']
```

```
In [66]: from sklearn.preprocessing import LabelEncoder
         test_data_two_class = pd.DataFrame({
             "sex" :  LabelEncoder().fit_transform(test_data_with_obj["sex"]),
             "income" :LabelEncoder().fit_transform(test_data_with_obj["income"]),
         })
```

```
In [67]: test_data_multiple_classes = pd.get_dummies(test_data_with_obj[test_multiple_classes])
         test_data_multiple_classes.head()
```

	workclass_Federal-gov	workclass_Local-gov	workclass_Never-worked	workclass_Private	workclass_Self-emp-inc	workclass_Self-emp-not-inc	workclass_State-gov	workclass_Without-pay	workclass_Not_avai
0	0	0	0	1	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0
2	0	0	0	1	0	0	0	0	0
3	0	0	0	0	0	0	0	0	1
4	0	0	0	1	0	0	0	0	0

5 rows × 52 columns

```
In [68]: test_lst = [test_data_two_class , test_data_multiple_classes]
         test_final_obj_data = pd.concat(test_lst , axis = 1)
         test_final_obj_data.head()
```

	sex	income	workclass_Federal-gov	workclass_Local-gov	workclass_Never-worked	workclass_Private	workclass_Self-emp-inc	workclass_Self-emp-not-inc	workclass_State-gov	workclass_Without-pay	...	
0	1	0	0	0	0	1	0	0	0	0	...	0
1	1	1	0	1	0	0	0	0	0	0	...	0
2	1	1	0	0	0	1	0	0	0	0	...	0
3	0	0	0	0	0	0	0	0	0	0	...	0
4	1	0	0	0	0	1	0	0	0	0	...	1

5 rows × 54 columns

```
In [69]: test_data_with_num.hours_per_week = test_data.hours_per_week/100
         test_data_with_num.age = test_data_with_num.age/100
         test_data_with_num.education_num = test_data_with_num.education_num/100

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py:5096: 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: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  self[name] = value
```

```
In [70]: from sklearn.preprocessing import MinMaxScaler
         sc = MinMaxScaler()
         test_lst_scaler_values = ["fnlwgt" ,"capital_gain" , "capital_loss"]
         test_data_array = sc.fit_transform(test_data_with_num[test_lst_scaler_values])
         test_data_array[:5]

array([[0.05167688, 0.        , 0.        ],
       [0.21901093, 0.        , 0.        ],
       [0.09941784, 0.07688077, 0.        ],
       [0.06094151, 0.        , 0.        ],
       [0.12539779, 0.        , 0.        ]])
```

```
In [71]: test_column_values = ["fnlwgt","capital_gain","capital_loss"]
         test_df = pd.DataFrame(data = test_data_array,
                          columns = test_column_values)
         test_df.tail()
```

	fnlwgt	capital_gain	capital_loss
16275	0.136723	0.000000	0.0
16276	0.208484	0.000000	0.0
16277	0.244762	0.000000	0.0
16278	0.047666	0.054551	0.0
16279	0.114195	0.000000	0.0

```
In [72]: test_lst_num_ =[test_data_with_num[["age","hours_per_week" , "education_num"]] , test_df]
         test_final_num_data = pd.concat(test_lst_num_ , axis = 1)
```

```
In [73]: test_final_df_lst = [test_final_num_data , test_final_obj_data]
         test_final_data = pd.concat(test_final_df_lst , axis = 1)
         test_final_data.head()
```

	age	hours_per_week	education_num	fnlwgt	capital_gain	capital_loss	sex	income	workclass_Federal-gov	workclass_Local-gov	...	
0	0.38	0.5	0.09	0.051677	0.000000	0.0	1	0	0	0	...	0
1	0.28	0.4	0.12	0.219011	0.000000	0.0	1	1	0	1	...	0
2	0.44	0.4	0.10	0.099418	0.076881	0.0	1	1	0	0	...	0
3	0.18	0.3	0.10	0.060942	0.000000	0.0	0	0	0	0	...	0
4	0.34	0.3	0.06	0.125398	0.000000	0.0	1	0	0	0	...	1

5 rows × 60 columns

```
In [74]: test_X = test_final_data.drop("income" ,axis =1)
         test_Y = test_final_data.income
```

```
In [75]: from sklearn.decomposition import PCA
         pca = PCA(n_components= 3)
         test_transformed_data = pca.fit_transform(test_X)
         test_transformed_data

array([[-0.99487154,  0.75211429, -0.11469905],
       [-1.04730551, -0.53059653,  0.17502619],
       [-0.79661605, -0.05466943, -0.36003945],
       ...,
       [-1.01022028, -0.33803172, -0.04831669],
       [ 0.49331387, -0.03596259,  0.05359316],
       [-1.14022288, -0.75833189,  0.23743085]])
```

Using Training Model For Testing

```
In [77]: from sklearn.tree import DecisionTreeClassifier
         dec = DecisionTreeClassifier(criterion='entropy' , max_depth= 10)
         dec

DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=10,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

```
In [80]: dec.fit(transformed_data , Y)

DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=10,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

```
In [81]: Y_pred =dec.predict(test_transformed_data)
```

```
In [83]: len(test_Y) , len(Y_pred)

(16280, 16280)
```

```
In [85]: MSE = np.square(np.subtract(test_Y,Y_pred)).mean()
         MSE

0.2516584766584767
```

```
In [88]: lst_pred = []
         for i in Y_pred:
             if i == 0:
                 lst_pred.append("<=50")
             else:
                 lst_pred.append(">50")
```

```
In [89]: lst_actual = []
         for i in test_Y:
             if i == 0:
                 lst_actual.append("<=50")
             else:
                 lst_actual.append(">50")
```

```
In [90]: final_pred =pd.DataFrame({
             "Y_actual" : lst_actual,
             "Y_pred" : lst_pred
         })
```

```
In [92]: final_pred.head()
```

	Y_actual	Y_pred
0	<=50	<=50
1	>50	>50
2	>50	<=50
3	<=50	<=50
4	<=50	<=50

```
In [ ]:
```