In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from IPython import get_ipython
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
data = pd.read_csv('adults.txt')
```

In [3]:

```
data.head()
```

Out[3]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	g
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Fe

In [4]:

```
data.tail()
```

Out[4]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	rac
32556	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	Whi
32557	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	Whi
32558	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	Whi
32559	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	Whi
32560	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	Whi

In [5]:

```
data.shape
```

Out[5]:

```
(32561, 15)
```

In [6]:

```
data.columns
```

Out[6]:

```
Index(['age', 'workclass', 'fnlwgt', 'education', 'educational-num',
       'marital-status', 'occupation', 'relationship', 'race', 'gender',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
       'income'],
      dtype='object')
```

In [7]:

```
data.duplicated().sum()
```

Out[7]:

```
24
```

In [8]:

```
data = data.drop_duplicates()
```

In [9]:

```
data.isnull().sum()
```

Out[9]:

```
age                0
workclass          0
fnlwgt             0
education          0
educational-num    0
marital-status     0
occupation         0
relationship       0
race               0
gender             0
capital-gain       0
capital-loss       0
hours-per-week     0
native-country     0
income             0
dtype: int64
```

In [10]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 32537 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   age              32537 non-null  int64 
 1   workclass        32537 non-null  object
 2   fnlwgt           32537 non-null  int64 
 3   education        32537 non-null  object
 4   educational-num  32537 non-null  int64 
 5   marital-status   32537 non-null  object
 6   occupation       32537 non-null  object
 7   relationship     32537 non-null  object
 8   race             32537 non-null  object
 9   gender           32537 non-null  object
 10  capital-gain     32537 non-null  int64 
 11  capital-loss     32537 non-null  int64 
 12  hours-per-week   32537 non-null  int64 
 13  native-country   32537 non-null  object
 14  income           32537 non-null  object
dtypes: int64(6), object(9)
memory usage: 4.0+ MB
```

In [11]:

```
data.describe()
```

Out[11]:

	age	fnlwgt	educational-num	capital-gain	capital-loss	hours-per-week
count	32537.000000	3.253700e+04	32537.000000	32537.000000	32537.000000	32537.000000
mean	38.585549	1.897808e+05	10.081815	1078.443741	87.368227	40.440329
std	13.637984	1.055565e+05	2.571633	7387.957424	403.101833	12.346889
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.369930e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

In [12]:

```
data.nunique()
```

Out[12]:

```
age                  73
workclass             9
fnlwgt            21648
education            16
educational-num      16
marital-status        7
occupation           15
relationship          6
race                  5
gender                2
capital-gain        119
capital-loss         92
hours-per-week       94
native-country       42
income                2
dtype: int64
```

In [13]:

```
data = data.drop(['fnlwgt', 'educational-num'], axis=1)
col_names = data.columns
for c in col_names:
    data = data.replace("?", np.NaN)
data = data.apply(lambda x: x.fillna(x.value_counts().index[0]))
```

In [15]:

```
from sklearn import preprocessing
```

In [16]:

```
data.replace(['Divorced', 'Married-AF-spouse',
             'Married-civ-spouse', 'Married-spouse-absent',
             'Never-married', 'Separated', 'Widowed'],
            ['divorced', 'married', 'married', 'married',
             'not married', 'not married', 'not married'],
            inplace=True)
category_col = ['workclass', 'race', 'education', 'marital-status',
                'occupation','relationship', 'gender',
                'native-country', 'income']
labelEncoder = preprocessing.LabelEncoder()
mapping_dict = {}
for col in category_col:
    data[col] = labelEncoder.fit_transform(data[col])

    le_name_mapping = dict(zip(labelEncoder.classes_,
                               labelEncoder.transform(labelEncoder.classes_)))
    mapping_dict[col] = le_name_mapping
print(mapping_dict)
```

```
{'workclass': {' ?': 0, ' Federal-gov': 1, ' Local-gov': 2, ' Never-worke
d': 3, ' Private': 4, ' Self-emp-inc': 5, ' Self-emp-not-inc': 6, ' State-
gov': 7, ' Without-pay': 8}, 'race': {' Amer-Indian-Eskimo': 0, ' Asian-Pa
c-Islander': 1, ' Black': 2, ' Other': 3, ' White': 4}, 'education': {' 10
th': 0, ' 11th': 1, ' 12th': 2, ' 1st-4th': 3, ' 5th-6th': 4, ' 7th-8th':
5, ' 9th': 6, ' Assoc-acdm': 7, ' Assoc-voc': 8, ' Bachelors': 9, ' Doctor
ate': 10, ' HS-grad': 11, ' Masters': 12, ' Preschool': 13, ' Prof-schoo
l': 14, ' Some-college': 15}, 'marital-status': {' Divorced': 0, ' Married
-AF-spouse': 1, ' Married-civ-spouse': 2, ' Married-spouse-absent': 3, ' N
ever-married': 4, ' Separated': 5, ' Widowed': 6}, 'occupation': {' ?': 0,
' Adm-clerical': 1, ' Armed-Forces': 2, ' Craft-repair': 3, ' Exec-manager
ial': 4, ' Farming-fishing': 5, ' Handlers-cleaners': 6, ' Machine-op-insp
ct': 7, ' Other-service': 8, ' Priv-house-serv': 9, ' Prof-specialty': 10,
' Protective-serv': 11, ' Sales': 12, ' Tech-support': 13, ' Transport-mov
ing': 14}, 'relationship': {' Husband': 0, ' Not-in-family': 1, ' Other-re
lative': 2, ' Own-child': 3, ' Unmarried': 4, ' Wife': 5}, 'gender': {' Fe
male': 0, ' Male': 1}, 'native-country': {' ?': 0, ' Cambodia': 1, ' Canad
a': 2, ' China': 3, ' Columbia': 4, ' Cuba': 5, ' Dominican-Republic': 6,
' Ecuador': 7, ' El-Salvador': 8, ' England': 9, ' France': 10, ' German
y': 11, ' Greece': 12, ' Guatemala': 13, ' Haiti': 14, ' Holand-Netherland
s': 15, ' Honduras': 16, ' Hong': 17, ' Hungary': 18, ' India': 19, ' Ira
n': 20, ' Ireland': 21, ' Italy': 22, ' Jamaica': 23, ' Japan': 24, ' Lao
s': 25, ' Mexico': 26, ' Nicaragua': 27, ' Outlying-US(Guam-USVI-etc)': 2
8, ' Peru': 29, ' Philippines': 30, ' Poland': 31, ' Portugal': 32, ' Puer
to-Rico': 33, ' Scotland': 34, ' South': 35, ' Taiwan': 36, ' Thailand': 3
7, ' Trinadad&Tobago': 38, ' United-States': 39, ' Vietnam': 40, ' Yugosla
via': 41}, 'income': {' <=50K': 0, ' >50K': 1}}
```

In [17]:

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```

In [18]:

```
X = data.values[:, 0:12]
Y = data.values[:, 12]
```

In [19]:

```
X_train, X_test, y_train, y_test = train_test_split(
            X, Y, test_size = 0.3, random_state = 100)

dt_clf_gini = DecisionTreeClassifier(criterion = "gini",
                                     random_state = 100,
                                     max_depth = 5,
                                     min_samples_leaf = 5)

dt_clf_gini.fit(X_train, y_train)
y_pred_gini = dt_clf_gini.predict(X_test)

print ("Decision Tree using Gini Index\nAccuracy is ",
             accuracy_score(y_test, y_pred_gini)*100 )
```

```
Decision Tree using Gini Index
Accuracy is  82.67772997336611
```

In [ ]:

```
<html>
<body>
    <h3>Income Prediction Form</h3>
<div>
<form action="/result" method="POST">
    <label for="age">Age</label>
    <input type="text" id="age" name="age">
    <br>
    <label for="w_class">Working Class</label>
    <select id="w_class" name="w_class">
    <option value="0">Federal-gov</option>
    <option value="1">Local-gov</option>
    <option value="2">Never-worked</option>
    <option value="3">Private</option>
    <option value="4">Self-emp-inc</option>
    <option value="5">Self-emp-not-inc</option>
    <option value="6">State-gov</option>
    <option value="7">Without-pay</option>
    </select>
    <br>
    <label for="edu">Education</label>
    <select id="edu" name="edu">
    <option value="0">10th</option>
    <option value="1">11th</option>
    <option value="2">12th</option>
    <option value="3">1st-4th</option>
    <option value="4">5th-6th</option>
    <option value="5">7th-8th</option>
    <option value="6">9th</option>
    <option value="7">Assoc-acdm</option>
    <option value="8">Assoc-voc</option>
    <option value="9">Bachelors</option>
    <option value="10">Doctorate</option>
    <option value="11">HS-grad</option>
    <option value="12">Masters</option>
    <option value="13">Preschool</option>
    <option value="14">Prof-school</option>
    <option value="15">16 - Some-college</option>
    </select>
    <br>
    <label for="martial_stat">Marital Status</label>
    <select id="martial_stat" name="martial_stat">
    <option value="0">divorced</option>
    <option value="1">married</option>
    <option value="2">not married</option>
    </select>
    <br>
    <label for="occup">Occupation</label>
    <select id="occup" name="occup">
    <option value="0">Adm-clerical</option>
    <option value="1">Armed-Forces</option>
    <option value="2">Craft-repair</option>
    <option value="3">Exec-managerial</option>
    <option value="4">Farming-fishing</option>
    <option value="5">Handlers-cleaners</option>
    <option value="6">Machine-op-inspect</option>
    <option value="7">Other-service</option>
    <option value="8">Priv-house-serv</option>
    <option value="9">Prof-specialty</option>
```

```
    <option value="10">Protective-serv</option>
    <option value="11">Sales</option>
    <option value="12">Tech-support</option>
    <option value="13">Transport-moving</option>
    </select>
    <br>
    <label for="relation">Relationship</label>
    <select id="relation" name="relation">
    <option value="0">Husband</option>
    <option value="1">Not-in-family</option>
    <option value="2">Other-relative</option>
    <option value="3">Own-child</option>
    <option value="4">Unmarried</option>
    <option value="5">Wife</option>
    </select>
    <br>
    <label for="race">Race</label>
    <select id="race" name="race">
    <option value="0">Amer Indian Eskimo</option>
    <option value="1">Asian Pac Islander</option>
    <option value="2">Black</option>
    <option value="3">Other</option>
    <option value="4">White</option>
    </select>
    <br>
    <label for="gender">Gender</label>
    <select id="gender" name="gender">
    <option value="0">Female</option>
    <option value="1">Male</option>
    </select>
    <br>
    <label for="c_gain">Capital Gain </label>
    <input type="text" id="c_gain" name="c_gain">btw:[0-99999]
    <br>
    <label for="c_loss">Capital Loss </label>
    <input type="text" id="c_loss" name="c_loss">btw:[0-4356]
    <br>
    <label for="hours_per_week">Hours per Week </label>
    <input type="text" id="hours_per_week" name="hours_per_week">btw:[1-99]
    <br>
    <label for="native-country">Native Country</label>
    <select id="native-country" name="native-country">
    <option value="0">Cambodia</option>
    <option value="1">Canada</option>
    <option value="2">China</option>
    <option value="3">Columbia</option>
    <option value="4">Cuba</option>
    <option value="5">Dominican Republic</option>
    <option value="6">Ecuador</option>
    <option value="7">El Salvadorr</option>
    <option value="8">England</option>
    <option value="9">France</option>
    <option value="10">Germany</option>
    <option value="11">Greece</option>
    <option value="12">Guatemala</option>
    <option value="13">Haiti</option>
    <option value="14">Netherlands</option>
    <option value="15">Honduras</option>
    <option value="16">HongKong</option>
    <option value="17">Hungary</option>
    <option value="18">India</option>
```

```
    <option value="19">Iran</option>
    <option value="20">Ireland</option>
    <option value="21">Italy</option>
    <option value="22">Jamaica</option>
    <option value="23">Japan</option>
    <option value="24">Laos</option>
    <option value="25">Mexico</option>
    <option value="26">Nicaragua</option>
    <option value="27">Outlying-US(Guam-USVI-etc)</option>
    <option value="28">Peru</option>
    <option value="29">Philippines</option>
    <option value="30">Poland</option>
    <option value="11">Portugal</option>
    <option value="32">Puerto-Rico</option>
    <option value="33">Scotland</option>
    <option value="34">South</option>
    <option value="35">Taiwan</option>
    <option value="36">Thailand</option>
    <option value="37">Trinadad&Tobago</option>
    <option value="38">United States</option>
    <option value="39">Vietnam</option>
    <option value="40">Yugoslavia</option>
    </select>
    <br>
    <input type="submit" value="Submit">
</form>
</div>
</body>
</html>
```

In [22]:

```
from flask import *
app = Flask(__name__)
```

In [23]:

```
# prediction function
def ValuePredictor(to_predict_list):
    to_predict = np.array(to_predict_list).reshape(1, 12)
    loaded_model = pickle.load(open("model.pkl", "rb"))
    result = loaded_model.predict(to_predict)
    return result[0]

@app.route('/result', methods = ['POST'])
def result():
    if request.method == 'POST':
        to_predict_list = request.form.to_dict()
        to_predict_list = list(to_predict_list.values())
        to_predict_list = list(map(int, to_predict_list))
        result = ValuePredictor(to_predict_list)
        if int(result)== 1:
            prediction ='Income more than 50K'
        else:
            prediction ='Income less that 50K'
        return render_template("result.html", prediction = prediction)

if __name__ == '__main__':
    app.run(debug = True)
```