Overview of the project

This aims at predicting the income of people based on 14 different features provided. Decision Trees was used for this proof of concept project

Cleaning the data and data preprocessing

- Checking for null value and string data type was done
- Null values were very less in number and hence were removed from the data
- Encoding was done and dummy variables were created
- At this point data was ready to be tested on the model

Implementing Decision Trees

- Default Hyperparametrs were used
- Accuracy of 84% was obtained

Hyper parameter tuning

- max_depth was iterated over the range [3,4,5,6,7,8,9], and min_sample_split was iterated over the range [2,3,4] and accuracy was calculated for all of them using GridSearchCV
- Best accuracy was found to at max_depth = 8 and min_sample_split = 4

In [39]:

```
import numpy as pd
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:

```
df = pd.read_csv('Decision.csv')
df.columns
```

In [165]:

```
df
```

Out[165]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relatio
0	90	?	77053	HS-grad	9	Widowed	?	Not-in-
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-
2	66	?	186061	Some-college	10	Widowed	?	Unm
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unm
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Owr
...	...	...	...	...	...	...	...	
32556	22	Private	310152	Some-college	10	Never-married	Protective-serv	Not-in-
32557	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	
32558	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Hu:
32559	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unm
32560	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Owr

32561 rows × 15 columns

In [166]:

```
%matplotlib inline
```

In [167]:

```
import warnings
warnings.filterwarnings("ignore")
```

In [168]:

```
df.describe()
```

Out[168]:

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.wee
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.0000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.4374
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.3474
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.0000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.0000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.0000
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.0000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.0000

In [169]:

```
df.head()
```

Out[169]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationshi
0	90	?	77053	HS-grad	9	Widowed	?	Not-in-famil
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-famil
2	66	?	186061	Some-college	10	Widowed	?	Unmarrie
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarrie
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-chil

In [170]:

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

Out[170]:

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

In [171]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
age               32561 non-null int64
workclass         32561 non-null object
fnlwgt            32561 non-null int64
education         32561 non-null object
education.num     32561 non-null int64
marital.status    32561 non-null object
occupation        32561 non-null object
relationship      32561 non-null object
race              32561 non-null object
sex               32561 non-null object
capital.gain      32561 non-null int64
capital.loss      32561 non-null int64
hours.per.week    32561 non-null int64
native.country    32561 non-null object
income            32561 non-null object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

In [172]:

```
df_miss = df[df.workclass == '?']
```

In [173]:

```
df_miss
```

Out[173]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relatio
0	90	?	77053	HS-grad	9	Widowed	?	Not-in-
2	66	?	186061	Some-college	10	Widowed	?	Unm
14	51	?	172175	Doctorate	16	Never-married	?	Not-in-
24	61	?	135285	HS-grad	9	Married-civ-spouse	?	Hu:
44	71	?	100820	HS-grad	9	Married-civ-spouse	?	Hu:
...	...	...	...	...	...	...	...	
32533	35	?	320084	Bachelors	13	Married-civ-spouse	?	
32534	30	?	33811	Bachelors	13	Never-married	?	Not-in-
32541	71	?	287372	Doctorate	16	Married-civ-spouse	?	Hu:
32543	41	?	202822	HS-grad	9	Separated	?	Not-in-
32544	72	?	129912	HS-grad	9	Married-civ-spouse	?	Hu:

1836 rows × 15 columns

In [174]:

```
df.workclass.unique()
```

Out[174]:

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

In [175]:

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

Out[175]:

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

In [176]:

```
df_categorical = df.select_dtypes(include = ['object'])

df_categorical.apply(lambda x: x=='?', axis= 0).sum()
```

Out[176]:

```
workclass         1836
education            0
marital.status       0
occupation        1843
relationship         0
race                 0
sex                  0
native.country     583
income               0
dtype: int64
```

In [177]:

```
df = df[df.workclass != '?']
```

In [178]:

```
df_categorical = df.select_dtypes(include = ['object'])

df_categorical.apply(lambda x: x=='?', axis= 0).sum()
```

Out[178]:

```
workclass           0
education           0
marital.status      0
occupation          7
relationship        0
race                0
sex                 0
native.country    556
income              0
dtype: int64
```

In [179]:

```
df = df[df.occupation != '?']
```

In [180]:

```
df = df[df['native.country'] != '?']
```

In [181]:

```
df
```

Out[181]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relatio
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unm
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Owr
5	34	Private	216864	HS-grad	9	Divorced	Other-service	Unm
6	38	Private	150601	10th	6	Separated	Adm-clerical	Unm
...	...	...	...	...	...	...	...	
32556	22	Private	310152	Some-college	10	Never-married	Protective-serv	Not-in-
32557	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	
32558	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Hu
32559	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unm
32560	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Owr

30162 rows × 15 columns

In [182]:

```
df_categorical.columns
```

Out[182]:

```
Index(['workclass', 'education', 'marital.status', 'occupation',
       'relationship', 'race', 'sex', 'native.country', 'income'],
      dtype='object')
```

In [183]:

```
x = pd.get_dummies(df[['workclass', 'education', 'marital.status', 'occupation',
       'relationship', 'race', 'sex', 'native.country']])
```

In [184]:

```
df.drop(['workclass', 'education', 'marital.status', 'occupation',
       'relationship', 'race', 'sex', 'native.country'], axis = 1, inplace = True)
```

In [185]:

```
df = pd.concat([df, x], axis = 1)
```

In [186]:

```
df.columns
```

Out[186]:

```
Index(['age', 'fnlwgt', 'education.num', 'capital.gain', 'capital.loss',
       'hours.per.week', 'income', 'workclass_Federal-gov',
       'workclass_Local-gov', 'workclass_Private',
       ...
       'native.country_Portugal', 'native.country_Puerto-Rico',
       'native.country_Scotland', 'native.country_South',
       'native.country_Taiwan', 'native.country_Thailand',
       'native.country_Trinadad&Tobago', 'native.country_United-States',
       'native.country_Vietnam', 'native.country_Yugoslavia'],
      dtype='object', length=105)
```

In [187]:

```
df['income'] = df['income'].astype('category')
```

In [192]:

```
df['income'] = pd.get_dummies(df['income'])['>50K']
```

In [ ]:

```

```

In [193]:

```
X = df.drop('income', axis = 1)
y = df['income']
```

In [194]:

```
df.columns
```

Out[194]:

```
Index(['age', 'fnlwgt', 'education.num', 'capital.gain', 'capital.loss',
       'hours.per.week', 'income', 'workclass_Federal-gov',
       'workclass_Local-gov', 'workclass_Private',
       ...
       'native.country_Portugal', 'native.country_Puerto-Rico',
       'native.country_Scotland', 'native.country_South',
       'native.country_Taiwan', 'native.country_Thailand',
       'native.country_Trinadad&Tobago', 'native.country_United-States',
       'native.country_Vietnam', 'native.country_Yugoslavia'],
      dtype='object', length=105)
```

In [ ]:

```

```

In [ ]:

```

```

In [195]:

```
from sklearn.model_selection import train_test_split
```

In [196]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4
2)
```

In [197]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [198]:

```
model = DecisionTreeClassifier(max_depth=5)
```

In [199]:

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

Out[199]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=5,
                       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 [200]:

```
prediction = model.predict(X_test)
```

In [203]:

```
from sklearn.metrics import classification_report, confusion_matrix
```

In [202]:

```
print(classification_report(prediction, y_test))
```

```
              precision    recall  f1-score   support

           0       0.95      0.85      0.90      5062
           1       0.50      0.78      0.61       971

    accuracy                           0.84      6033
   macro avg       0.73      0.81      0.75      6033
weighted avg       0.88      0.84      0.85      6033
```

In [205]:

```
print(confusion_matrix(prediction, y_test))
```

```
[[4315  747]
 [ 218  753]]
```

In [210]:

```
(4315+753)/(4315+747+753+218)
```

Out[210]:

```
0.840046411403945
```

In [ ]:

```

```

In [ ]:

```
from sklearn.model_selection import GridSearchCV
model = DecisionTreeClassifier()

param_grid = {'max_depth':[3,4,5,6,7,8,9], 'min_samples_split': [2,3,4]}
```

In [222]:

```
optimise = GridSearchCV(model, param_grid, cv=10)

optimise.fit(X_train, y_train)
```

In [224]:

```
optimise.best_estimator_
```

Out[224]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=8,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=4,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

In [225]:

```
optimise.best_score_
```

Out[225]:

```
0.8540760081230055
```

In [226]:

```
best_model = DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=8,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=4,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

In [229]:

```
best_model.fit(X_train, y_train)

print(classification_report(best_model.predict(X_test), y_test))
```

```
              precision    recall  f1-score   support

           0       0.95      0.86      0.90      5006
           1       0.54      0.78      0.64      1027

    accuracy                           0.85      6033
   macro avg       0.74      0.82      0.77      6033
weighted avg       0.88      0.85      0.86      6033
```

Thanks a lot

Do connect on linkedin: https://www.linkedin.com/in/jay-dhanwant-71926a167/ (https://www.linkedin.com/in/jay-dhanwant-71926a167/)

In [ ]:

```

```