In [1]:

```
!pip install category_encoders
import category_encoders as ce
import pandas as pd
from sklearn import datasets
import numpy as np
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
import warnings
warnings.filterwarnings("ignore")
```

```
Requirement already satisfied: category_encoders in c:\users\acer\anaconda3\lib\site-packages (2.2.2)
Requirement already satisfied: scikit-learn>=0.20.0 in c:\users\acer\anaconda3\lib\site-packages (from ca
tegory_encoders) (0.23.2)
Requirement already satisfied: patsy>=0.5.1 in c:\users\acer\anaconda3\lib\site-packages (from category_e
ncoders) (0.5.1)
Requirement already satisfied: pandas>=0.21.1 in c:\users\acer\anaconda3\lib\site-packages (from category
_encoders) (1.1.3)
Requirement already satisfied: statsmodels>=0.9.0 in c:\users\acer\anaconda3\lib\site-packages (from cate
gory_encoders) (0.12.0)
Requirement already satisfied: scipy>=1.0.0 in c:\users\acer\anaconda3\lib\site-packages (from category_e
ncoders) (1.5.2)
Requirement already satisfied: numpy>=1.14.0 in c:\users\acer\anaconda3\lib\site-packages (from category_
encoders) (1.19.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\acer\anaconda3\lib\site-packages (from sc
ikit-learn>=0.20.0->category_encoders) (2.1.0)
Requirement already satisfied: joblib>=0.11 in c:\users\acer\anaconda3\lib\site-packages (from scikit-lea
rn>=0.20.0->category_encoders) (0.17.0)
Requirement already satisfied: six in c:\users\acer\anaconda3\lib\site-packages (from patsy>=0.5.1->categ
ory_encoders) (1.15.0)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\acer\anaconda3\lib\site-packages (from
pandas>=0.21.1->category_encoders) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in c:\users\acer\anaconda3\lib\site-packages (from pandas>=0.
21.1->category_encoders) (2020.1)
```

In [2]:

```
# import fraud check set
fc = pd.read_csv('/Users/acer/Sandesh Pal/Data Science Assgn/random Forest/Fraud_check.csv')
```

In [3]:

```
fc
```

Out[3]:

	Undergrad	Marital.Status	Taxable.Income	City.Population	Work.Experience	Urban
0	NO	Single	68833	50047	10	YES
1	YES	Divorced	33700	134075	18	YES
2	NO	Married	36925	160205	30	YES
3	YES	Single	50190	193264	15	YES
4	NO	Married	81002	27533	28	NO
...	...	...	...	...	...	...
595	YES	Divorced	76340	39492	7	YES
596	YES	Divorced	69967	55369	2	YES
597	NO	Divorced	47334	154058	0	YES
598	YES	Married	98592	180083	17	NO
599	NO	Divorced	96519	158137	16	NO

600 rows × 6 columns

In [4]:

```
# checking for null values
fc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 600 entries, 0 to 599
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Undergrad        600 non-null    object
 1   Marital.Status   600 non-null    object
 2   Taxable.Income   600 non-null    int64 
 3   City.Population  600 non-null    int64 
 4   Work.Experience  600 non-null    int64 
 5   Urban            600 non-null    object
dtypes: int64(3), object(3)
memory usage: 28.2+ KB
```

In [5]:

```
fc.describe()
```

Out[5]:

	Taxable.Income	City.Population	Work.Experience
count	600.000000	600.000000	600.000000
mean	55208.375000	108747.368333	15.558333
std	26204.827597	49850.075134	8.842147
min	10003.000000	25779.000000	0.000000
25%	32871.500000	66966.750000	8.000000
50%	55074.500000	106493.500000	15.000000
75%	78611.750000	150114.250000	24.000000
max	99619.000000	199778.000000	30.000000

In [6]:

```
import category_encoders as ce
# encode variables with ordinal encoding
encoder = ce.OrdinalEncoder(cols=['Undergrad', 'Marital.Status', 'Urban'])
fc1 = encoder.fit_transform(fc)
```

In [8]:

```
# Converting the Target column i.e. Taxable Income into Categorical value
tax_val = []
for value in fc["Taxable.Income"]:
    if value<=30000:
        tax_val.append("Risky")
    else:
        tax_val.append("Good")

fc1["tax_val"]= tax_val
```

In [9]:

```
fc1
```

Out[9]:

	Undergrad	Marital.Status	Taxable.Income	City.Population	Work.Experience	Urban	tax_val
0	1	1	68833	50047	10	1	Good
1	2	2	33700	134075	18	1	Good
2	1	3	36925	160205	30	1	Good
3	2	1	50190	193264	15	1	Good
4	1	3	81002	27533	28	2	Good
...	...	...	...	...	...	...	...
595	2	2	76340	39492	7	1	Good
596	2	2	69967	55369	2	1	Good
597	1	2	47334	154058	0	1	Good
598	2	3	98592	180083	17	2	Good
599	1	2	96519	158137	16	2	Good

600 rows × 7 columns

In [10]:

```
x = fc1.drop(['tax_val','Taxable.Income'], axis =1)
y = fc1['tax_val']
```

In [11]:

```
x.head()
```

Out[11]:

	Undergrad	Marital.Status	City.Population	Work.Experience	Urban
0	1	1	50047	10	1
1	2	2	134075	18	1
2	1	3	160205	30	1
3	2	1	193264	15	1
4	1	3	27533	28	2

In [12]:

```
y.value_counts()
```

Out[12]:

```
Good     476
Risky    124
Name: tax_val, dtype: int64
```

Random Forest Classification

In [13]:

```
num_trees = 100
max_features = 4
kfold = KFold(n_splits=20)
model = RandomForestClassifier(n_estimators=num_trees, max_features=max_features)
results = cross_val_score(model, x, y, cv=kfold)
print(results.mean())
```

```
0.7500000000000001
```

lets use the various ensemble techniques to check the accuracy %

Bagging

In [15]:

```
# Bagged Decision Trees for Classification
from sklearn.ensemble import BaggingClassifier
seed = 7
kfold = KFold(n_splits=20, random_state=seed)
cart = DecisionTreeClassifier()
num_trees = 100
model = BaggingClassifier(base_estimator=cart, n_estimators=num_trees, random_state=seed)
results = cross_val_score(model, x, y, cv=kfold)
print(results.mean())
```

```
0.7350000000000001
```

Boosting

In [16]:

```
# AdaBoost Classification
from sklearn.ensemble import AdaBoostClassifier
num_trees = 100
seed=7
kfold = KFold(n_splits=20, random_state=seed)
model = AdaBoostClassifier(n_estimators=num_trees, random_state=seed)
results = cross_val_score(model, x, y, cv=kfold)
print(results.mean())
```

```
0.7733333333333335
```

Stacking

In [17]:

```
# Stacking Ensemble for Classification
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
```

```python
from sklearn.ensemble import VotingClassifier
```

In [18]:

```python
# create the sub models
estimators = []
model1 = LogisticRegression(max_iter=500)
estimators.append(('logistic', model1))
model2 = DecisionTreeClassifier()
estimators.append(('cart', model2))
model3 = SVC()
estimators.append(('svm', model3))
# create the ensemble model
ensemble = VotingClassifier(estimators)
results = cross_val_score(ensemble, x, y, cv=kfold)
print(results.mean())
```

```
0.7933333333333334
```

In [ ]: