In [1]:
```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
fraud=pd.read_csv("/Users/SAURABH/Saurabh patil/DATA SCIENCE/random Forest/Fraud_check.csv")
fraud
```

Out[2]:

	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 [3]:
```
#Changing the categorical variables into dummies.
df = pd.get_dummies(fraud)

#Converting the Target variable i.e. Taxable Income into Categorical (As mentioned in the problem statement)
df['Category'] = pd.cut(df['Taxable.Income'],
                        bins=[0,30000, np.inf],
                        labels=['Risky','Good'],
                        include_lowest=True)
df
```

Out[3]:

	Taxable.Income	City.Population	Work.Experience	Undergrad_NO	Undergrad_YES	Marital.Status_Divorced	Marital.Status_Married	Marital.Status_Single	Urban_NO	Urban_YES	Category
0	68833	50047	10	1	0	0	0	1	0	1	Good
1	33700	134075	18	0	1	1	0	0	0	1	Good
2	36925	160205	30	1	0	0	1	0	0	1	Good
3	50190	193264	15	0	1	0	0	1	0	1	Good
4	81002	27533	28	1	0	0	1	0	1	0	Good
...	...	...	...	...	...	...	...	...	...	...	...
595	76340	39492	7	0	1	1	0	0	0	1	Good
596	69967	55369	2	0	1	1	0	0	0	1	Good
597	47334	154058	0	1	0	1	0	0	0	1	Good
598	98592	180083	17	0	1	0	1	0	1	0	Good
599	96519	158137	16	1	0	1	0	0	1	0	Good

600 rows × 11 columns

In [4]:
```
# Random Forest Classification
from pandas import read_csv
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
```

In [5]:
```
array = df.values
X = array[:,1:10]
Y = array[:,10]
```

In [6]:
```
num_trees = 100
max_features = 3
kfold = KFold(n_splits=10, random_state=7)
model = RandomForestClassifier(n_estimators=num_trees, max_features=max_features)
results = cross_val_score(model, X, Y, cv=kfold)
print(results.mean())
```

```
0.735
```

Let's try if we can increase thee cv score using ensemble techniques

Bagging

In [7]:
```
from pandas import read_csv
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier

seed = 7

cart = DecisionTreeClassifier()
model1 = BaggingClassifier(base_estimator=cart, n_estimators=num_trees, random_state=seed)
results1 = cross_val_score(model1, X, Y, cv=kfold)
print(results1.mean()*100)
```

```
73.0
```

Boosting

In [8]:
```
# AdaBoost Classification
from pandas import read_csv
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import AdaBoostClassifier

model2 = AdaBoostClassifier(n_estimators=num_trees, random_state=seed)
results2 = cross_val_score(model2, X, Y, cv=kfold)
print(results2.mean()*100)
```

```
77.33333333333331
```

Stacking

In [9]:
```
# Stacking Ensemble for Classification
from pandas import read_csv
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier
```

Iteration-1

In [11]:
```
# create the sub models
estimators = []
model3 = LogisticRegression(max_iter=500)
estimators.append(('logistic', model3))
model4 = DecisionTreeClassifier()
estimators.append(('cart', model4))
model5 = SVC()
estimators.append(('svm', model5))
model6 = BaggingClassifier(base_estimator=cart, n_estimators=num_trees, random_state=seed)
estimators.append(('bagging', model6))
model7 = AdaBoostClassifier(n_estimators=num_trees, random_state=seed)
estimators.append(('boosting', model7))

# create the ensemble modelIter
ensemble = VotingClassifier(estimators)
results3 = cross_val_score(ensemble, X, Y, cv=kfold)
print(results3.mean()*100)
```

```
78.66666666666666
```

Iteration-2

In [12]:
```
# create the sub models
estimators = []
model8 = LogisticRegression(max_iter=500)
estimators.append(('logistic', model8))
model9 = DecisionTreeClassifier()
estimators.append(('cart', model9))
model10 = BaggingClassifier(base_estimator=cart, n_estimators=num_trees, random_state=seed)
estimators.append(('bagging', model10))
model11 = AdaBoostClassifier(n_estimators=num_trees, random_state=seed)
estimators.append(('boosting', model11))

# create the ensemble model
ensemble = VotingClassifier(estimators)
results4 = cross_val_score(ensemble, X, Y, cv=kfold)
print(results4.mean()*100)
```

```
78.66666666666666
```

Iteration-3

In [13]:
```
# create the sub models
estimators = []
model12 = LogisticRegression(max_iter=500)
estimators.append(('logistic', model12))
model13 = DecisionTreeClassifier()
estimators.append(('cart', model13))
model14 = AdaBoostClassifier(n_estimators=num_trees, random_state=seed)
estimators.append(('boosting', model14))

# create the ensemble modSel
ensemble = VotingClassifier(estimators)
results5 = cross_val_score(ensemble, X, Y, cv=kfold)
print(results5.mean()*100)
```

```
78.49999999999999
```

Iteration-4

In [14]:
```
# create the sub models
estimators = []
model15 = DecisionTreeClassifier()
estimators.append(('cart', model15))
model16 = AdaBoostClassifier(n_estimators=num_trees, random_state=seed)
estimators.append(('boosting', model16))

# create the ensemble model
ensemble = VotingClassifier(estimators)
results6 = cross_val_score(ensemble, X, Y, cv=kfold)
print(results6.mean()*100)
```

```
78.49999999999999
```

Iteration-5

In [16]:
```
# create the sub models
estimators = []
model17 = LogisticRegression(max_iter=500)
estimators.append(('logistic', model17))
model18 = AdaBoostClassifier(n_estimators=num_trees, random_state=seed)
estimators.append(('boosting', model18))

# create the ensemble model
ensemble = VotingClassifier(estimators)
results6 = cross_val_score(ensemble, X, Y, cv=kfold)
print(results6.mean()*100)
```

```
79.33333333333333
```

Iteration-6

In [17]:
```
# create the sub models
estimators = []
model19 = BaggingClassifier(base_estimator=cart, n_estimators=num_trees, random_state=seed)
estimators.append(('bagging', model19))
model20 = AdaBoostClassifier(n_estimators=num_trees, random_state=seed)
estimators.append(('boosting', model20))

# create the ensemble model
ensemble = VotingClassifier(estimators)
results7 = cross_val_score(ensemble, X, Y, cv=kfold)
print(results7.mean()*100)
```

```
78.49999999999999
```

Since the cv score for iteration 5 was the max, so we can consider it to be our final model

In [ ]:
```
```