

```
In [1]: import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier
from sklearn import preprocessing
```

```
In [2]: data=pd.read_csv('Fraud_check.csv')
data
```

```
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

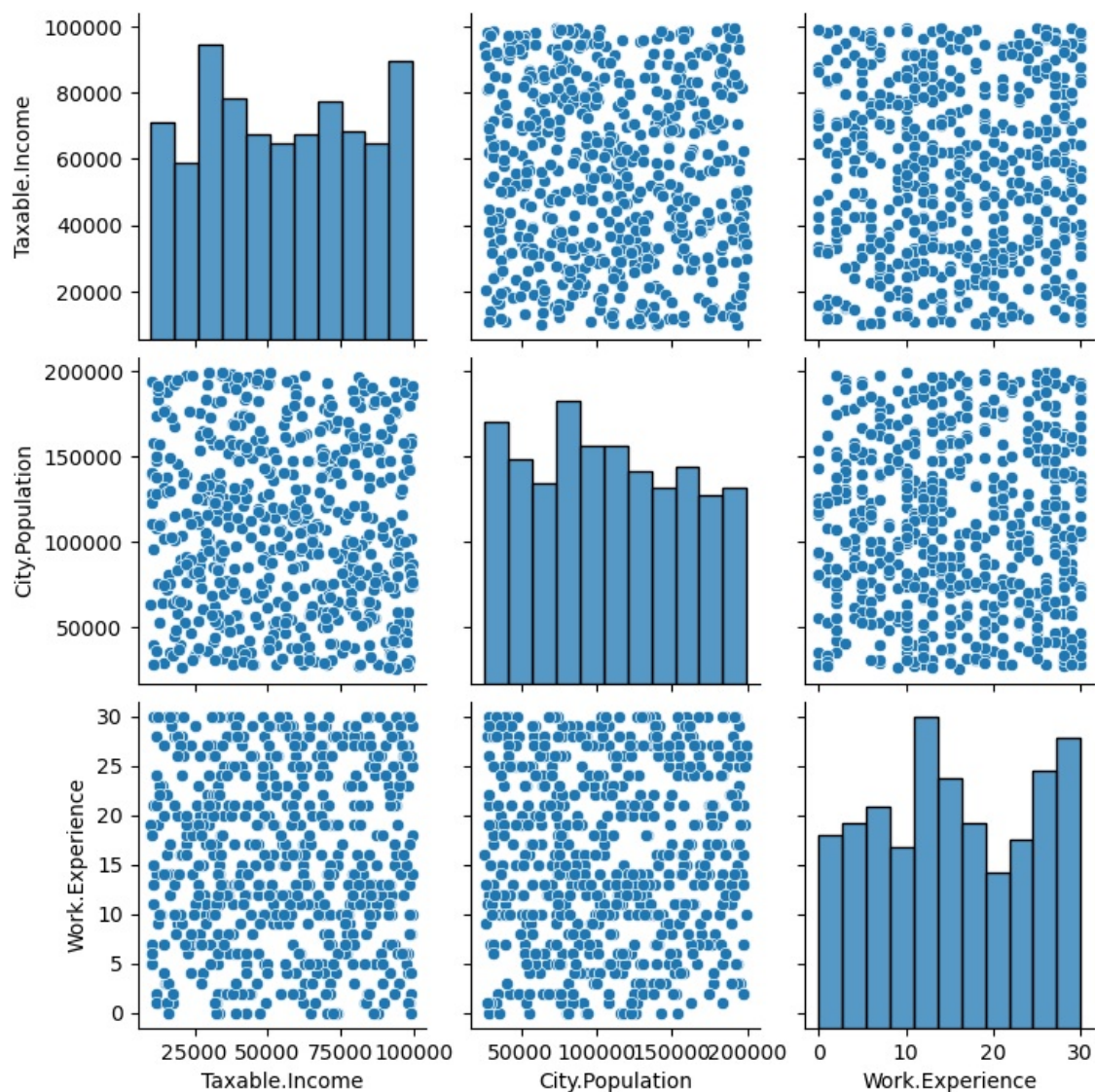
600 rows × 6 columns

```
In [3]: data.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 [4]: sns.pairplot(data)
```

```
Out[4]: <seaborn.axisgrid.PairGrid at 0x2a112711490>
```



```
In [6]: sns.heatmap(data.isnull(),cmap='Reds')
```

```
Out[6]: <AxesSubplot:>
```

```
In [7]: plt.figure(figsize=(20,10))
sns.heatmap(data.corr(),annot=True)
```

```
Out[7]: <AxesSubplot:>
```

```
In [8]: label_encoder = preprocessing.LabelEncoder()
data['Undergrad']= label_encoder.fit_transform(data['Undergrad'])
data['Urban']= label_encoder.fit_transform(data['Urban'])
data['Marital.Status']= label_encoder.fit_transform(data['Marital.Status'])
```

```
In [9]: data
```

Out[9]:	Undergrad	Marital.Status	Taxable.Income	City.Population	Work.Experience	Urban
0	0	2	68833	50047	10	1
1	1	0	33700	134075	18	1
2	0	1	36925	160205	30	1
3	1	2	50190	193264	15	1
4	0	1	81002	27533	28	0
...
595	1	0	76340	39492	7	1
596	1	0	69967	55369	2	1
597	0	0	47334	154058	0	1
598	1	1	98592	180083	17	0
599	0	0	96519	158137	16	0

600 rows × 6 columns

```
In [10]: data['Status'] = data['Taxable.Income'].apply(lambda Income: 'Risky' if Income <= 30000 else 'Good')
```

```
In [11]: data['Status'] = label_encoder.fit_transform(data['Status'])
```

```
In [12]: data
```

Out[12]:	Undergrad	Marital.Status	Taxable.Income	City.Population	Work.Experience	Urban	Status
0	0	2	68833	50047	10	1	0
1	1	0	33700	134075	18	1	0
2	0	1	36925	160205	30	1	0
3	1	2	50190	193264	15	1	0
4	0	1	81002	27533	28	0	0
...
595	1	0	76340	39492	7	1	0
596	1	0	69967	55369	2	1	0
597	0	0	47334	154058	0	1	0
598	1	1	98592	180083	17	0	0
599	0	0	96519	158137	16	0	0

600 rows × 7 columns

```
In [13]: data.Status.unique()
```

```
Out[13]: array([0, 1])
```

```
In [14]: x=data.iloc[:,0:4]
y=data['Status']
```

```
In [15]: num_trees = 100
seed=8
kfold = KFold(n_splits=100, shuffle = True, random_state=seed)
model = AdaBoostClassifier(n_estimators=num_trees, random_state=seed)
results = cross_val_score(model, x,y, cv=kfold)
print(results.mean())

0.9983333333333334
```

```
In [16]: kfold = KFold(n_splits=10,shuffle=True, random_state=8)
estimators = []
model1 = LogisticRegression(max_iter=100)
estimators.append(('logistic', model1))
model2 = DecisionTreeClassifier()
estimators.append(('cart', model2))
model3 = SVC()
estimators.append(('svm', model3))
ensemble = VotingClassifier(estimators)
results = cross_val_score(ensemble, x, y, cv=kfold)
print(results.mean())

0.9833333333333332
```

```
In [17]: num_trees = 100
max_features = 3
```

```
kfold = KFold(n_splits=10, shuffle= True ,random_state=8)
model = RandomForestClassifier(n_estimators=num_trees, max_features=max_features)
results = cross_val_score(model, x, y, cv=kfold)
print(results.mean())
```

0.9983333333333334

```
In [18]: num_trees = 100
seed=8
kfold = KFold(n_splits=100, shuffle = True, random_state=seed)
model = AdaBoostClassifier(n_estimators=num_trees, random_state=seed)
results = cross_val_score(model, x,y, cv=kfold)
print(results.mean())
```

0.9983333333333334

In []:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js