In [35]:

```
import pandas as pd
import numpy as np
from pandas_profiling import ProfileReport
from scipy.stats import pearsonr
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import GridSearchCV
# from xgboost import XGBClassifier
from sklearn.ensemble import AdaBoostClassifier
#from catboost import CatBoostRegressor
```

In [5]:

```
!pip install xgboost
```

```
^C
```

In [16]:

```
df = pd.read_csv('adult.csv')
```

In [137…

```
df
```

Out[137…

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

32561 rows × 15 columns

In [138…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             32561 non-null  int64 
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64 
 3   education       32561 non-null  object
 4   education-num   32561 non-null  int64 
 5   marital-status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital-gain    32561 non-null  int64 
 11  capital-loss    32561 non-null  int64 
 12  hours-per-week  32561 non-null  int64 
 13  country         32561 non-null  object
 14  salary          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

In [50]:

```
df.describe()
```

Out[50]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429
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	2.370510e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

In [8]:

```
pf =ProfileReport(df)
```

In [9]:

```
pf.to_widgets()
```

In [15]:

```
display(df.corr().abs())
```

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week
age	1.000000	0.076646	0.036527	0.077674	0.057775	0.068756
fnlwgt	0.076646	1.000000	0.043195	0.000432	0.010252	0.018768
education-num	0.036527	0.043195	1.000000	0.122630	0.079923	0.148123
capital-gain	0.077674	0.000432	0.122630	1.000000	0.031615	0.078409
capital-loss	0.057775	0.010252	0.079923	0.031615	1.000000	0.054256
hours-per-week	0.068756	0.018768	0.148123	0.078409	0.054256	1.000000

In [14]:

```
(df.values.astype(str)=='?').sum()
```

Out[14]:

```
0
```

In [20]:

```
n= df.shape[0]
n
```

Out[20]:

```
32561
```

In [17]:

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
df['workclass'] = le.fit_transform(df['workclass'])
df['marital-status'] = le.fit_transform(df['marital-status'])
df['occupation'] = le.fit_transform(df['occupation'])
df['relationship'] = le.fit_transform(df['relationship'])
df['race'] = le.fit_transform(df['race'])
df['sex'] = le.fit_transform(df['sex'])
df['country'] = le.fit_transform(df['country'])
df['salary'] = le.fit_transform(df['salary'])
df['education'] = le.fit_transform(df['education'])

df.head()
```

Out[17]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	se
0	39	7	77516	9	13	4	1	1	4	
1	50	6	83311	9	13	2	4	0	4	
2	38	4	215646	11	9	0	6	1	4	
3	53	4	234721	1	7	2	6	0	2	
4	28	4	338409	9	13	2	10	5	2	

In [18]:

```
x = df.drop(['salary'], axis = 1)
y = df['salary']
```

In [19]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.2,
```

In [147…

```
from sklearn.naive_bayes import GaussianNB
gb = GaussianNB()
gb.fit(x_train,y_train)
```

Out[147…

```
GaussianNB()
```

In [36]:

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

In [149…

```
y_pred = gb.predict(x_test)
print(classification_report(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
print(accuracy_score(y_test,y_pred)*100)
```

```
              precision    recall  f1-score   support

           0       0.81      0.94      0.87      4920
           1       0.64      0.32      0.43      1593

    accuracy                           0.79      6513
   macro avg       0.73      0.63      0.65      6513
weighted avg       0.77      0.79      0.76      6513

[[4635  285]
 [1082  511]]
79.01120835252571
```

In [121…

```
rf = RandomForestClassifier()
rf.fit(x_train,y_train)
```

Out[121…

```
RandomForestClassifier()
```

In [122…

```
y_pred = rf.predict(x_test)
print(classification_report(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
print(accuracy_score(y_test,y_pred)*100)
```

```
              precision    recall  f1-score   support

           0       0.89      0.93      0.91      4920
           1       0.74      0.63      0.68      1593

    accuracy                           0.86      6513
   macro avg       0.81      0.78      0.79      6513
weighted avg       0.85      0.86      0.85      6513

[[4564  356]
 [ 587 1006]]
85.52126516198373
```

In [186…

```
param_grid = {
    'n_estimators': [350,400],
    'max_features': ['auto', 'sqrt'],
    'max_depth' : [7,8,9,10],
    'criterion' :['gini', 'entropy']
}
```

In [ ]:

```
CV_rfc = GridSearchCV(estimator=rf, param_grid=param_grid, cv= 3)
CV_rfc.fit(x_train, y_train)
```

In [155…

```
CV_rfc.best_params_
```

Out[155…

```
{'criterion': 'gini',
 'max_depth': 7,
 'max_features': 'sqrt',
 'n_estimators': 120}
```

In [221…

```
rfc1=RandomForestClassifier(random_state=42, max_features='sqrt', n_estima
```

In [222…

```
rfc1.fit(x_train, y_train)
```

Out[222…

```
RandomForestClassifier(max_depth=22, max_features='sqrt', n_estimators=320,
                       random_state=42)
```

In [223…

```
pred=rfc1.predict(x_test)
```

In [225…

```
print("Accuracy for Random Forest on CV data: ",accuracy_score(y_test,pred
```

```
Accuracy for Random Forest on CV data:  86.3196683555965
```

In [20]:

```
ab_clf = AdaBoostClassifier(random_state=42)
```

In [21]:

```
parameters = {
    'n_estimators': [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 20],
    'learning_rate': [(0.97 + x / 100) for x in range(0, 8)],
    'algorithm': ['SAMME', 'SAMME.R']
}
```

In [24]:

```
clf = GridSearchCV(ab_clf, parameters, cv=5,n_jobs=-1)
clf.fit(x_train, y_train)
```

Out[24]:

```
GridSearchCV(cv=5, estimator=AdaBoostClassifier(random_state=42), n_jobs=-
1,
             param_grid={'algorithm': ['SAMME', 'SAMME.R'],
                         'learning_rate': [0.97, 0.98, 0.99, 1.0, 1.01, 1.0
2,
                                           1.03, 1.04],
                         'n_estimators': [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 1
2,
                                          20]})
```

In [26]:

```
clf.best_params_
```

Out[26]:

```
{'algorithm': 'SAMME.R', 'learning_rate': 0.98, 'n_estimators': 20}
```

In [50]:

```
ab_clf1 = AdaBoostClassifier(random_state=42,algorithm= 'SAMME.R', learnin
```

In [51]:

```
ab_clf1.fit(x_train, y_train)
```

Out[51]:

```
AdaBoostClassifier(learning_rate=0.98, n_estimators=320, random_state=42)
```

In [48]:

```
predd=ab_clf1.predict(x_test)
```

In [49]:

```
print("Accuracy for AdaBoostClassifier on CV data: ",accuracy_score(y_test
```

```
Accuracy for AdaBoostClassifier on CV data:  86.78028558268079
```

In [ ]: