Q1) Prepare a classification model using SVM for salary data

Data Description:

Age -- age of a person

Workclass -- A work class is a grouping of work

Education-- Education of an individuals

Maritalstatus -- Marital status of an individulas

Occupation -- occupation of an individuals

Relationship --

Race -- Race of an Individual

Sex -- Gender of an Individual

Capitalgain -- profit received from the sale of an investment

Capitalloss -- A decrease in the value of a capital asset

Hoursperweek -- number of hours work per week

Native -- Native of an individual

Salary -- salary of an individual

In [1]:
```
# Import the data
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.model_selection import GridSearchCV
```

In [2]:
```
train = pd.read_csv('SalaryData_Train.csv')
train
```

Out[2]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex	capitalgain	capitalloss	hoursperweek	native	Salary
0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
30156	27	Private	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	38	United-States	<=50K
30157	40	Private	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40	United-States	>50K
30158	58	Private	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	40	United-States	<=50K
30159	22	Private	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	0	20	United-States	<=50K
30160	52	Self-emp-inc	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	0	40	United-States	>50K

30161 rows × 14 columns

In [3]:
```
test = pd.read_csv('SalaryData_Test.csv')
test
```

Out[3]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex	capitalgain	capitalloss	hoursperweek	native	Salary
0	25	Private	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	United-States	<=50K
1	38	Private	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States	<=50K
2	28	Local-gov	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States	>50K
3	44	Private	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	United-States	>50K
4	34	Private	10th	6	Never-married	Other-service	Not-in-family	White	Male	0	0	30	United-States	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
15055	33	Private	Bachelors	13	Never-married	Prof-specialty	Own-child	White	Male	0	0	40	United-States	<=50K
15056	39	Private	Bachelors	13	Divorced	Prof-specialty	Not-in-family	White	Female	0	0	36	United-States	<=50K
15057	38	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	0	50	United-States	<=50K
15058	44	Private	Bachelors	13	Divorced	Adm-clerical	Own-child	Asian-Pac-Islander	Male	5455	0	40	United-States	<=50K
15059	35	Self-emp-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	60	United-States	>50K

15060 rows × 14 columns

In [4]:
```
train.head()
```

Out[4]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex	capitalgain	capitalloss	hoursperweek	native	Salary
0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

In [5]:
```
test.head()
```

Out[5]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex	capitalgain	capitalloss	hoursperweek	native	Salary
0	25	Private	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	United-States	<=50K
1	38	Private	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States	<=50K
2	28	Local-gov	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States	>50K
3	44	Private	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	United-States	>50K
4	34	Private	10th	6	Never-married	Other-service	Not-in-family	White	Male	0	0	30	United-States	<=50K

In [6]:
```
train.shape
```

Out[6]: (30161, 14)

In [7]:
```
test.shape
```

Out[7]: (15060, 14)

In [8]:
```
1 train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30161 entries, 0 to 30160
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   age            30161 non-null  int64 
 1   workclass      30161 non-null  object
 2   education      30161 non-null  object
 3   educationno    30161 non-null  int64 
 4   maritalstatus  30161 non-null  object
 5   occupation     30161 non-null  object
 6   relationship   30161 non-null  object
 7   race           30161 non-null  object
 8   sex            30161 non-null  object
 9   capitalgain    30161 non-null  int64 
 10  capitalloss    30161 non-null  int64 
 11  hoursperweek   30161 non-null  int64 
 12  native         30161 non-null  object
 13  Salary         30161 non-null  object
dtypes: int64(5), object(9)
memory usage: 3.2+ MB
```

In [9]:
```
1 test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15060 entries, 0 to 15059
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   age            15060 non-null  int64 
 1   workclass      15060 non-null  object
 2   education      15060 non-null  object
 3   educationno    15060 non-null  int64 
 4   maritalstatus  15060 non-null  object
 5   occupation     15060 non-null  object
 6   relationship   15060 non-null  object
 7   race           15060 non-null  object
 8   sex            15060 non-null  object
 9   capitalgain    15060 non-null  int64 
 10  capitalloss    15060 non-null  int64 
 11  hoursperweek   15060 non-null  int64 
 12  native         15060 non-null  object
 13  Salary         15060 non-null  object
dtypes: int64(5), object(9)
memory usage: 1.6+ MB
```

In [10]:
```
1 train.describe()
```

Out[10]:

	age	educationno	capitalgain	capitalloss	hoursperweek
count	30161.000000	30161.000000	30161.000000	30161.000000	30161.000000
mean	38.438115	10.121316	1092.044064	88.302311	40.931269
std	13.134830	2.550037	7406.466611	404.121321	11.980182
min	17.000000	1.000000	0.000000	0.000000	1.000000
25%	28.000000	9.000000	0.000000	0.000000	40.000000
50%	37.000000	10.000000	0.000000	0.000000	40.000000
75%	47.000000	13.000000	0.000000	0.000000	45.000000
max	90.000000	16.000000	99999.000000	4356.000000	99.000000

In [11]:
```
1 test.describe()
```

Out[11]:

	age	educationno	capitalgain	capitalloss	hoursperweek
count	15060.000000	15060.000000	15060.000000	15060.000000	15060.000000
mean	38.768327	10.112749	1120.301594	89.041899	40.951594
std	13.380676	2.558727	7703.181842	406.283245	12.062831
min	17.000000	1.000000	0.000000	0.000000	1.000000
25%	28.000000	9.000000	0.000000	0.000000	40.000000
50%	37.000000	10.000000	0.000000	0.000000	40.000000
75%	48.000000	13.000000	0.000000	0.000000	45.000000
max	90.000000	16.000000	99999.000000	3770.000000	99.000000

In [12]:
```
train.corr()
```

Out[12]:

	age	educationno	capitalgain	capitalloss	hoursperweek
age	1.000000	0.043525	0.080152	0.060278	0.101598
educationno	0.043525	1.000000	0.124416	0.079691	0.152522
capitalgain	0.080152	0.124416	1.000000	-0.032218	0.080431
capitalloss	0.060278	0.079691	-0.032218	1.000000	0.052454
hoursperweek	0.101598	0.152522	0.080431	0.052454	1.000000

In [13]:
```
test.corr()
```

Out[13]:

	age	educationno	capitalgain	capitalloss	hoursperweek
age	1.000000	0.026123	0.078760	0.057745	0.102758
educationno	0.026123	1.000000	0.131750	0.085817	0.133691
capitalgain	0.078760	0.131750	1.000000	-0.031876	0.090501
capitalloss	0.057745	0.085817	-0.031876	1.000000	0.057712
hoursperweek	0.102758	0.133691	0.090501	0.057712	1.000000

Data Preprocessing

In [14]:
```
lb = LabelEncoder()
```

In [15]:
```
train["workclass"] = lb.fit_transform(train["workclass"])
train["education"] = lb.fit_transform(train["education"])
train["maritalstatus"] = lb.fit_transform(train["maritalstatus"])
train["occupation"] = lb.fit_transform(train["occupation"])
train["relationship"] = lb.fit_transform(train["relationship"])
train["race"] = lb.fit_transform(train["race"])
train["sex"] = lb.fit_transform(train["sex"])
train["native"] = lb.fit_transform(train["native"])
train["Salary"] = lb.fit_transform(train["Salary"])
```

In [16]:
```
test["workclass"] = lb.fit_transform(test["workclass"])
test["education"] = lb.fit_transform(test["education"])
test["maritalstatus"] = lb.fit_transform(test["maritalstatus"])
test["occupation"] = lb.fit_transform(test["occupation"])
test["relationship"] = lb.fit_transform(test["relationship"])
test["race"] = lb.fit_transform(test["race"])
test["sex"] = lb.fit_transform(test["sex"])
test["native"] = lb.fit_transform(test["native"])
test["Salary"] = lb.fit_transform(test["Salary"])
```

EDA

In [17]:
```
train = train.iloc[: 2000, :]
```

In [18]:
```
train.info()
```

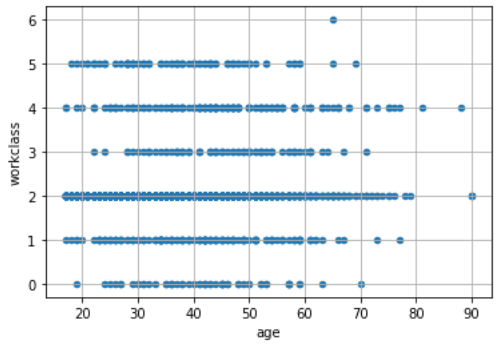
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   age            2000 non-null   int64
 1   workclass      2000 non-null   int32
 2   education      2000 non-null   int32
 3   educationno    2000 non-null   int64
 4   maritalstatus  2000 non-null   int32
 5   occupation     2000 non-null   int32
 6   relationship   2000 non-null   int32
 7   race           2000 non-null   int32
 8   sex            2000 non-null   int32
 9   capitalgain    2000 non-null   int64
 10  capitalloss    2000 non-null   int64
 11  hoursperweek   2000 non-null   int64
 12  native         2000 non-null   int32
 13  Salary         2000 non-null   int32
dtypes: int32(9), int64(5)
memory usage: 148.6 KB
```

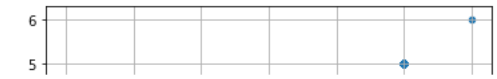
In [19]:
```
test = test.iloc[: 1300, :]
```

In [20]:
```
test.info()
```

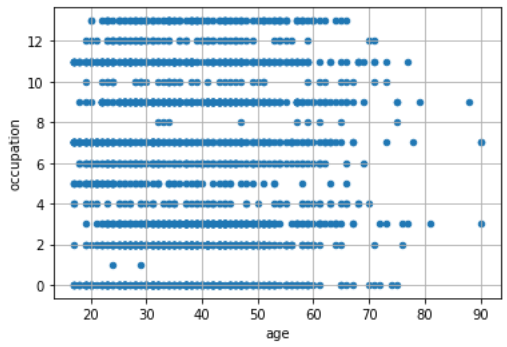
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1300 entries, 0 to 1299
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             1300 non-null   int64
 1   workclass       1300 non-null   int32
 2   education       1300 non-null   int32
 3   educationno     1300 non-null   int64
 4   maritalstatus   1300 non-null   int32
 5   occupation      1300 non-null   int32
 6   relationship    1300 non-null   int32
 7   race            1300 non-null   int32
 8   sex             1300 non-null   int32
 9   capitalgain     1300 non-null   int64
 10  capitalloss     1300 non-null   int64
 11  hoursperweek    1300 non-null   int64
 12  native          1300 non-null   int32
 13  Salary          1300 non-null   int32
dtypes: int32(9), int64(5)
memory usage: 96.6 KB
```

In [21]:
```
for i in train.describe().columns[:-2]:
    train.plot.scatter(i,'workclass', grid=True)
```

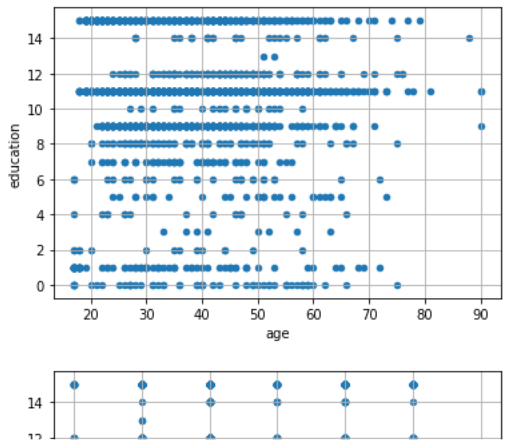




In [22]:
```
for i in train.describe().columns[:-2]:
    train.plot.scatter(i,'occupation',grid=True)
```



In [23]:
```
for i in train.describe().columns[:-2]:
    train.plot.scatter(i,'education',grid=True)
```



In [24]:
```
train.corr()
```

Out[24]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex	capitalgain	capitalloss	hoursperweek
age	1.000000	0.080136	-0.004007	0.014781	-0.249467	-0.004634	-0.216588	0.015168	0.050730	0.081112	0.058997	0.114429
workclass	0.080136	1.000000	0.029167	0.068866	-0.043219	0.033209	-0.074186	0.074418	0.087332	0.038314	-0.003069	0.031221
education	-0.004007	0.029167	1.000000	0.328746	-0.047668	-0.028564	-0.033833	0.031098	-0.004879	0.035363	0.012082	0.060470
educationno	0.014781	0.068866	0.328746	1.000000	-0.062303	0.098459	-0.091217	0.075867	0.034123	0.095804	0.062601	0.172302
maritalstatus	-0.249467	-0.043219	-0.047668	-0.062303	1.000000	0.075036	0.157226	-0.083280	-0.078456	-0.044395	-0.016550	-0.187437
occupation	-0.004634	0.033209	-0.028564	0.098459	0.075036	1.000000	-0.065478	0.035830	0.072483	0.016453	-0.015165	0.032509
relationship	-0.216588	-0.074186	-0.033833	-0.091217	0.157226	-0.065478	1.000000	-0.100663	-0.557999	-0.052849	-0.042243	-0.256052
race	0.015168	0.074418	0.031098	0.075867	-0.083280	0.035830	-0.100663	1.000000	0.076281	0.018565	0.043238	0.065593
sex	0.050730	0.087332	-0.004879	0.034123	-0.078456	0.072483	-0.557999	0.076281	1.000000	0.030977	0.049696	0.205762
capitalgain	0.081112	0.038314	0.035363	0.095804	-0.044395	0.016453	-0.052849	0.018565	0.030977	1.000000	-0.033968	0.064462
capitalloss	0.058997	-0.003069	0.012082	0.062601	-0.016550	-0.015165	-0.042243	0.043238	0.049696	-0.033968	1.000000	0.056110
hoursperweek	0.114429	0.031221	0.060470	0.172302	-0.187437	0.032509	-0.256052	0.065593	0.205762	0.064462	0.056110	1.000000
native	-0.001914	-0.036263	0.085718	0.057602	0.002006	-0.000106	-0.054397	0.155655	0.029781	-0.028919	-0.018007	-0.003643
Salary	0.231176	0.064561	0.051282	0.308324	-0.199289	0.026793	-0.211663	0.068448	0.182859	0.220183	0.135484	0.225754

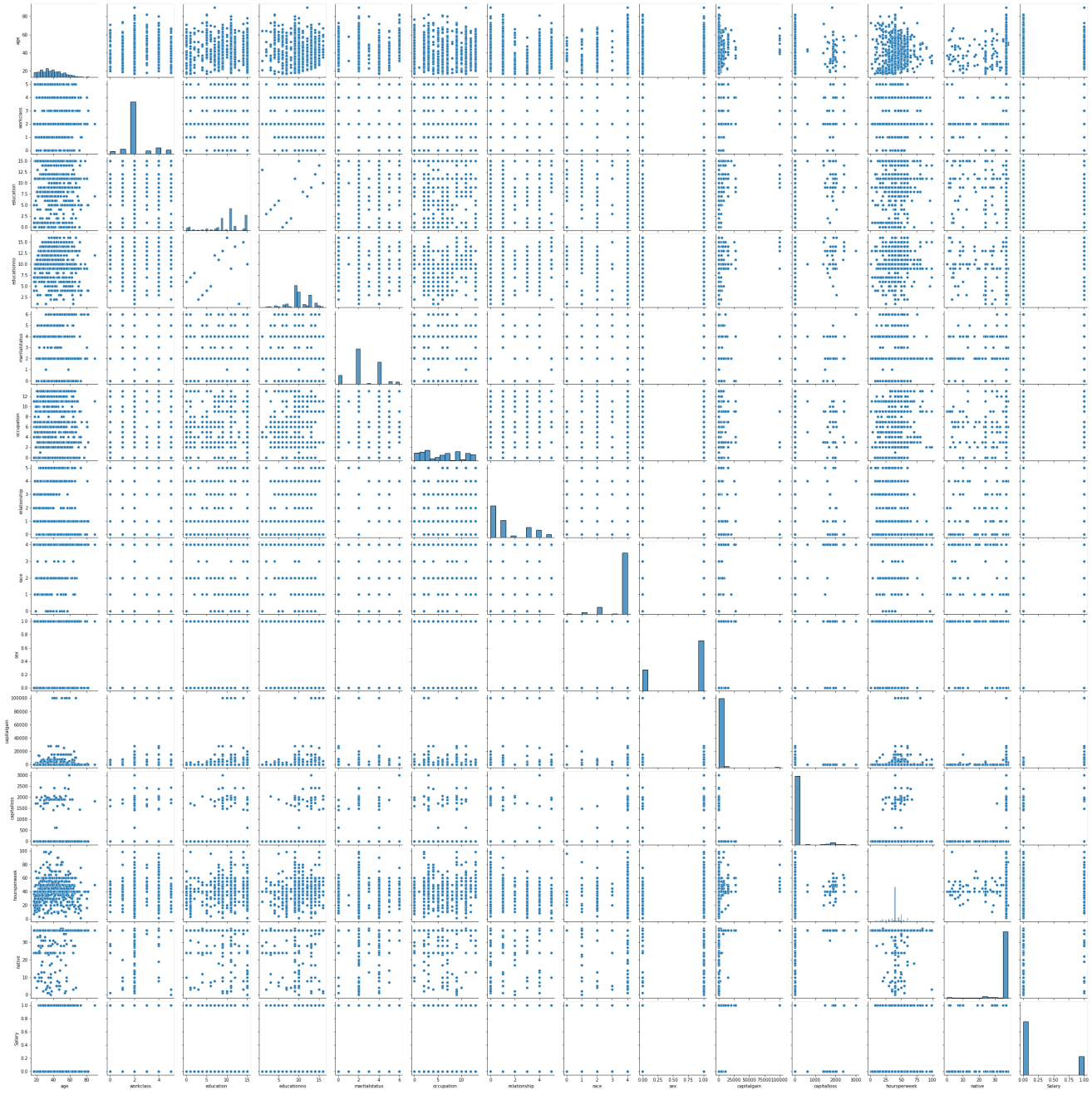
In [25]:
```
test.corr()
```

Out[25]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex	capitalgain	capitalloss	hoursperweek
age	1.000000	0.132698	-0.058456	-0.028238	-0.299098	-0.035747	-0.253358	0.003103	0.074865	0.107361	0.057418	0.106105
workclass	0.132698	1.000000	0.068824	0.066220	-0.060726	0.011914	-0.082095	0.065081	0.049687	0.053677	0.062026	0.077106
education	-0.058456	0.068824	1.000000	0.409201	-0.039852	-0.023030	-0.024427	-0.005501	-0.011597	0.060160	0.038734	0.063598
educationno	-0.028238	0.066220	0.409201	1.000000	-0.091147	0.077972	-0.050679	0.072135	-0.011529	0.170971	0.111713	0.145317
maritalstatus	-0.299098	-0.060726	-0.039852	-0.091147	1.000000	-0.012545	0.197796	-0.085572	-0.147099	-0.071843	-0.018926	-0.161722
occupation	-0.035747	0.011914	-0.023030	0.077972	-0.012545	1.000000	-0.063657	0.037160	0.073262	0.004247	-0.038230	0.018785
relationship	-0.253358	-0.082095	-0.024427	-0.050679	0.197796	-0.063657	1.000000	-0.158517	-0.588370	-0.060524	-0.037262	-0.298815
race	0.003103	0.065081	-0.005501	0.072135	-0.085572	0.037160	-0.158517	1.000000	0.072954	0.017319	0.064287	0.058017
sex	0.074865	0.049687	-0.011597	-0.011529	-0.147099	0.073262	-0.588370	0.072954	1.000000	0.070784	0.053486	0.233998
capitalgain	0.107361	0.053677	0.060160	0.170971	-0.071843	0.004247	-0.060524	0.017319	0.070784	1.000000	-0.036077	0.128418
capitalloss	0.057418	0.062026	0.038734	0.111713	-0.018926	-0.038230	-0.037262	0.064287	0.053486	-0.036077	1.000000	0.062200
hoursperweek	0.106105	0.077106	0.063598	0.145317	-0.161722	0.018785	-0.298815	0.058017	0.233998	0.128418	0.062200	1.000000
native	0.018697	0.043338	0.069760	0.106144	0.005488	0.004193	0.015321	0.160356	-0.037813	0.032191	0.051320	-0.026799
Salary	0.226220	0.077615	0.119016	0.313422	-0.226209	0.018910	-0.259728	0.076017	0.206553	0.249281	0.163200	0.231284

In [26]:
```
1 sns.pairplot(test)
```

Out[26]: <seaborn.axisgrid.PairGrid at 0x20c90ab47c0>



Data Splitting

In [27]:
```
X_train=train.iloc[:,:-1]
X_train
```

Out[27]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex	capitalgain	capitalloss	hoursperweek	native
0	39	5	9	13	4	0	1	4	1	2174	0	40	37
1	50	4	9	13	2	3	0	4	1	0	0	13	37
2	38	2	11	9	0	5	1	4	1	0	0	40	37
3	53	2	1	7	2	5	0	2	1	0	0	40	37
4	28	2	9	13	2	9	5	2	0	0	0	40	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1995	33	2	11	9	5	10	3	4	0	0	0	40	37
1996	41	2	11	9	2	6	0	4	1	0	0	40	37
1997	51	2	6	5	2	13	0	4	1	0	0	40	37
1998	42	2	11	9	2	11	0	4	1	0	0	48	37
1999	27	2	9	13	2	9	5	4	0	0	0	40	37

2000 rows × 13 columns

In [28]:
```
y_train=train.iloc[:,-1]
y_train
```

Out[28]:
```
0       0
1       0
2       0
3       0
4       0
       ..
1995    0
1996    0
1997    0
1998    0
1999    1
Name: Salary, Length: 2000, dtype: int32
```

In [29]:
```
X_test=test.iloc[:,:-1]
X_test
```

Out[29]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex	capitalgain	capitalloss	hoursperweek	native
0	25	2	1	7	4	6	3	2	1	0	0	40	37
1	38	2	11	9	2	4	0	4	1	0	0	50	37
2	28	1	7	12	2	10	0	4	1	0	0	40	37
3	44	2	15	10	2	6	0	2	1	7688	0	40	37
4	34	2	0	6	4	7	1	4	1	0	0	30	37
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1295	66	4	15	10	2	13	0	2	1	0	0	60	37
1296	40	2	15	10	4	2	2	0	1	0	0	30	37
1297	37	2	4	3	2	6	0	4	1	0	0	40	7
1298	34	2	9	13	2	11	0	4	1	0	0	40	37
1299	41	4	12	14	6	9	1	4	0	0	0	20	37

1300 rows × 13 columns

In [30]:
```
y_test=test.iloc[:,-1]
y_test
```

Out[30]:
```
0       0
1       0
2       1
3       1
4       0
       ..
1295    0
1296    0
1297    0
1298    0
1299    0
Name: Salary, Length: 1300, dtype: int32
```

In [31]:
```
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

Out[31]: ((2000, 13), (2000,), (1300, 13), (1300,))

SVM Model

In [32]:
```
model = SVC()

model.fit(X_train, y_train)
```

Out[32]: SVC()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Predicting model

In [33]:
```
y_pred = model.predict(X_test)
y_pred
```

Out[33]: array([0, 0, 0, ..., 0, 0, 0])

Model Evaluation

In [34]:
```
print(confusion_matrix(y_test, y_pred))
```

```
[[961   5]
 [267  67]]
```

In [35]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.78      0.99      0.88       966
           1       0.93      0.20      0.33       334

    accuracy                           0.79      1300
   macro avg       0.86      0.60      0.60      1300
weighted avg       0.82      0.79      0.74      1300
```

Inference

- Model is not predicting well, so we will improve the model by hyperparameter tunning using grid search method.

Improving Model using Grid Search CV

In [36]:
```
param_grid = {'C' : [1, 5, 10, 15, 20], 'gamma' : [1, 0.1, 0.01, 0.001, 0.0001], 'kernel' : ['rbt']}
```

In [37]:
```
grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3, cv = 5)
```

In [38]:
```
1 grid.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 25 candidates, totalling 125 fits
[CV 1/5] END .....C=1, gamma=1, kernel=rbt;, score=nan total time=   0.0s
[CV 2/5] END .....C=1, gamma=1, kernel=rbt;, score=nan total time=   0.0s
[CV 3/5] END .....C=1, gamma=1, kernel=rbt;, score=nan total time=   0.0s
[CV 4/5] END .....C=1, gamma=1, kernel=rbt;, score=nan total time=   0.0s
[CV 5/5] END .....C=1, gamma=1, kernel=rbt;, score=nan total time=   0.0s
[CV 1/5] END .....C=1, gamma=0.1, kernel=rbt;, score=nan total time=   0.0s
[CV 2/5] END .....C=1, gamma=0.1, kernel=rbt;, score=nan total time=   0.0s
[CV 3/5] END .....C=1, gamma=0.1, kernel=rbt;, score=nan total time=   0.0s
[CV 4/5] END .....C=1, gamma=0.1, kernel=rbt;, score=nan total time=   0.0s
[CV 5/5] END .....C=1, gamma=0.1, kernel=rbt;, score=nan total time=   0.0s
[CV 1/5] END .....C=1, gamma=0.01, kernel=rbt;, score=nan total time=   0.0s
[CV 2/5] END .....C=1, gamma=0.01, kernel=rbt;, score=nan total time=   0.0s
[CV 3/5] END .....C=1, gamma=0.01, kernel=rbt;, score=nan total time=   0.0s
[CV 4/5] END .....C=1, gamma=0.01, kernel=rbt;, score=nan total time=   0.0s
[CV 5/5] END .....C=1, gamma=0.01, kernel=rbt;, score=nan total time=   0.0s
[CV 1/5] END .....C=1, gamma=0.001, kernel=rbt;, score=nan total time=   0.0s
[CV 2/5] END .....C=1, gamma=0.001, kernel=rbt;, score=nan total time=   0.0s
[CV 3/5] END .....C=1, gamma=0.001, kernel=rbt;, score=nan total time=   0.0s
```

In [40]:
```
1 from sklearn.model_selection import GridSearchCV
```

In [44]:
```
1 print(grid.param_grid)
```

```
{'C': [1, 5, 10, 15, 20], 'gamma': [1, 0.1, 0.01, 0.001, 0.0001], 'kernel': ['rbt']}
```

In [59]:
```
1 grid.estimator
```

Out[59]:
```
SVC()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [ ]:
```
1
```