Import Necessary Libraries

In [1]:
```
import pandas as pd
from sklearn import metrics
import seaborn as sns
from sklearn.svm import SVC
from matplotlib import pyplot as plt
from sklearn.preprocessing import LabelEncoder
```

In [2]:
```
test_data=pd.read_csv('SalaryData_Test(1).csv',sep=',')
train_data=pd.read_csv('SalaryData_Train(1).csv',sep=',')
```

In [3]:
```
salary_data=test_data.append(train_data)
```

In [4]:
```
test=test_data.copy()
train=train_data.copy()
```

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

Out[5]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race
0	25	Private	11th	7	Never-married	Machine-op-inspct	Own-child	Black
1	38	Private	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White
2	28	Local-gov	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White
3	44	Private	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black
4	34	Private	10th	6	Never-married	Other-service	Not-in-family	White

In [6]:
```
train.head()
```

Out[6]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race
0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White
1	50	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White
2	38	Private	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White
3	53	Private	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black
4	28	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black

In [7]:
```
str_c = ["workclass","education","maritalstatus","occupation","relation
```

In [8]:
```
number = LabelEncoder()
```

In [9]:
```
for i in str_c:
    train[i]= number.fit_transform(train[i])
    test[i]=number.fit_transform(test[i])
```

In [10]:
```
test.head()
```

Out[10]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race
0	25	2	1	7	4	6	3	2
1	38	2	11	9	2	4	0	4
2	28	1	7	12	2	10	0	4
3	44	2	15	10	2	6	0	2
4	34	2	0	6	4	7	1	4

In [11]:
```
train.head()
```

Out[11]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race
0	39	5	9	13	4	0	1	4
1	50	4	9	13	2	3	0	4
2	38	2	11	9	0	5	1	4
3	53	2	1	7	2	5	0	2
4	28	2	9	13	2	9	5	2

In [12]:
```
mapping = {' >50K': 1, ' <=50K': 2}
```

In [13]:
```
train = train.replace({'Salary': mapping})
test = test.replace({'Salary': mapping})
```

In [15]:
```
salary_data=train.append(test)
```

In [17]:
```
salary=salary_data.copy()
salary.head()
```

Out[17]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race
0	39	5	9	13	4	0	1	4
1	50	4	9	13	2	3	0	4
2	38	2	11	9	0	5	1	4
3	53	2	1	7	2	5	0	2
4	28	2	9	13	2	9	5	2

In [18]:
```
salary.shape
```

Out[18]: (45221, 14)

In [20]:
```
salary.describe().T
```

Out[20]:

	count	mean	std	min	25%	50%	75%	max
age	45221.0	38.548086	13.217981	17.0	28.0	37.0	47.0	90.0
workclass	45221.0	2.204507	0.958132	0.0	2.0	2.0	2.0	6.0
education	45221.0	10.313217	3.816992	0.0	9.0	11.0	12.0	15.0
educationno	45221.0	10.118463	2.552909	1.0	9.0	10.0	13.0	16.0
maritalstatus	45221.0	2.585148	1.500460	0.0	2.0	2.0	4.0	6.0
occupation	45221.0	5.969572	4.026444	0.0	2.0	6.0	9.0	13.0
relationship	45221.0	1.412684	1.597242	0.0	0.0	1.0	3.0	5.0
race	45221.0	3.680281	0.832361	0.0	4.0	4.0	4.0	4.0
sex	45221.0	0.675062	0.468357	0.0	0.0	1.0	1.0	1.0
capitalgain	45221.0	1101.454700	7506.511295	0.0	0.0	0.0	0.0	99999.0
capitalloss	45221.0	88.548617	404.838249	0.0	0.0	0.0	0.0	4356.0
hoursperweek	45221.0	40.938038	12.007640	1.0	40.0	40.0	45.0	99.0
native	45221.0	35.431503	5.931380	0.0	37.0	37.0	37.0	39.0
Salary	45221.0	1.752151	0.431769	1.0	2.0	2.0	2.0	2.0

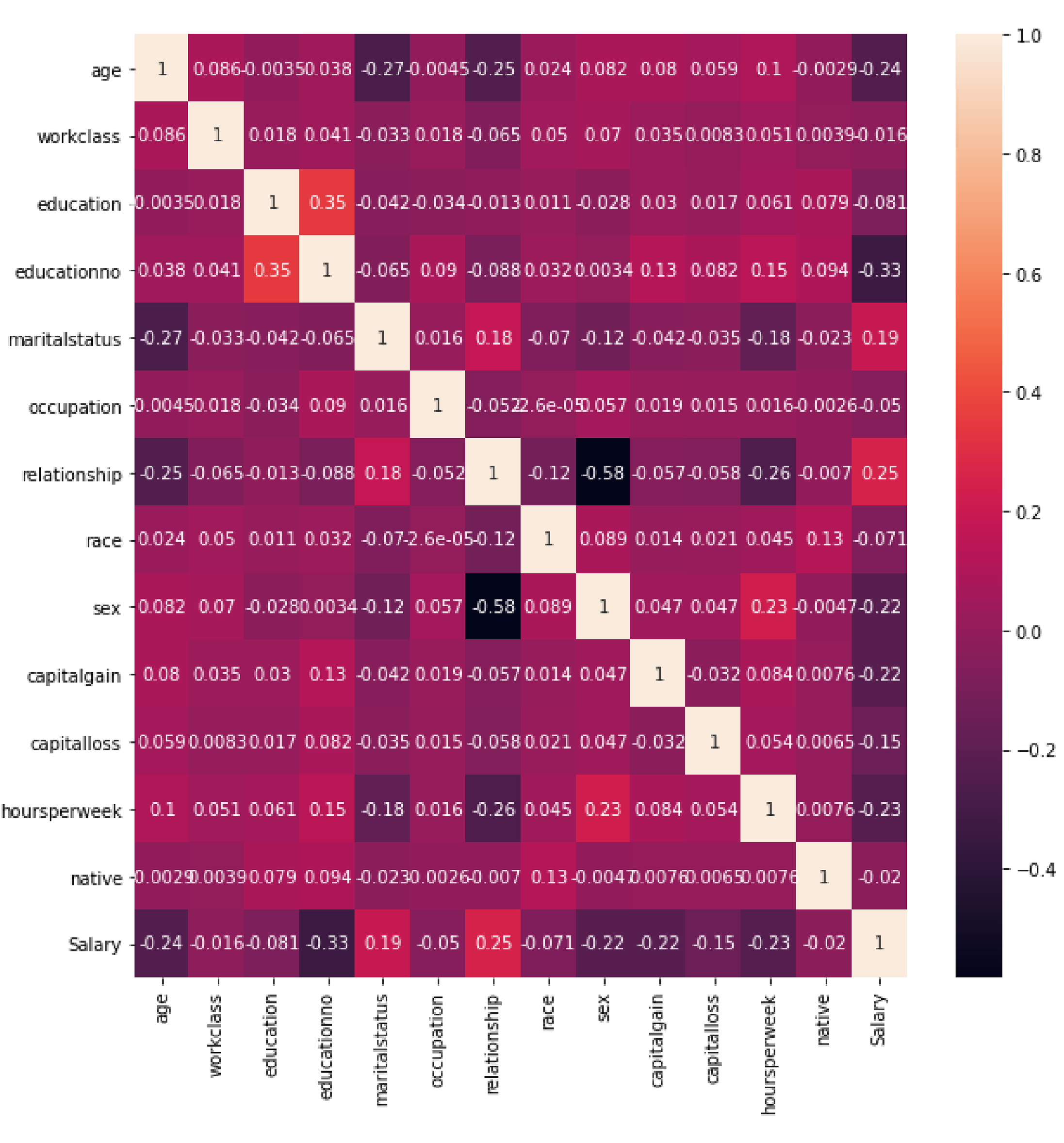
In [19]:
```
salary.isna().sum()
```

Out[19]:
```
age              0
workclass        0
education        0
educationno      0
maritalstatus    0
occupation       0
relationship     0
race             0
sex              0
capitalgain      0
capitalloss      0
hoursperweek     0
native           0
Salary           0
dtype: int64
```

In [20]:
```
corr = salary.corr()
```

In [21]:
```
plt.figure(figsize=(10,10))
sns.heatmap(corr,annot=True)
```

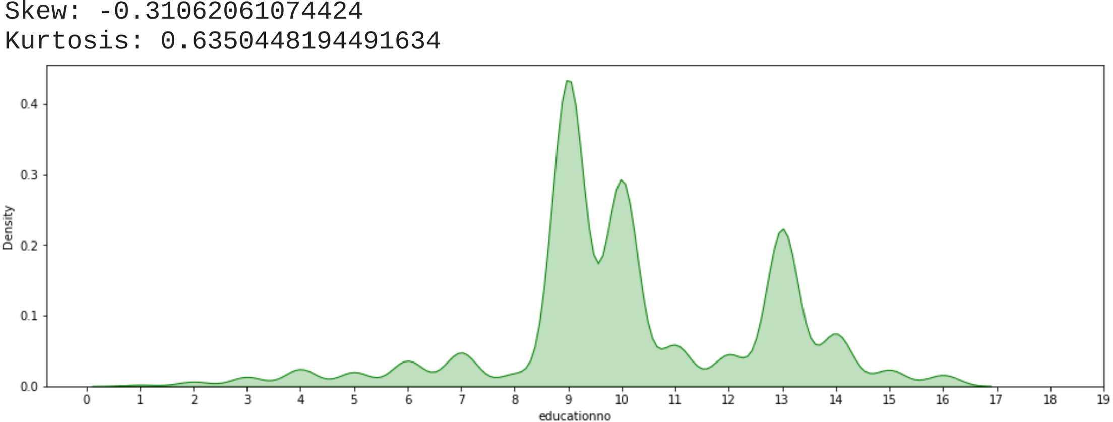
Out[21]: <AxesSubplot:>



In [22]:
```
plt.rcParams["figure.figsize"] = 9,5
```

In [24]:
```
plt.figure(figsize=(16,5))
print("Skew: {}".format(salary['educationno'].skew()))
print("Kurtosis: {}".format(salary['educationno'].kurtosis()))
ax = sns.kdeplot(salary['educationno'],shade=True,color='g')
plt.xticks([i for i in range(0,20,1)])
plt.show()
```

```
Skew: -0.31062061074424
Kurtosis: 0.6350448194491634
```



The Data is nagatively skewed and has low kurtosis value

In [33]:
```
dfa = salary[salary.columns[0:10]]
obj_colum = dfa.select_dtypes(include='object').columns.tolist()
```

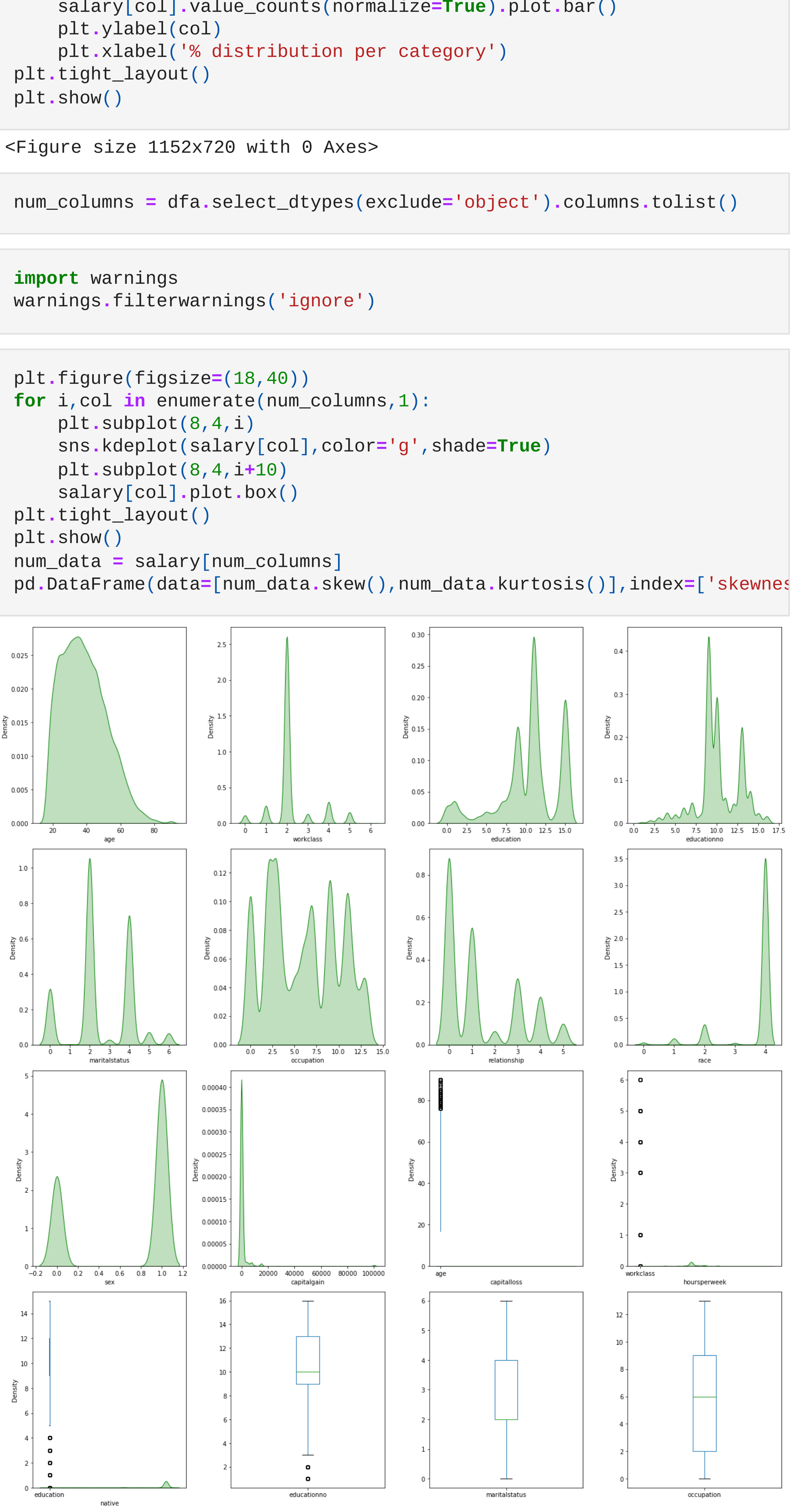
In [34]:
```
plt.figure(figsize=(16,10))
for i,col in enumerate(obj_colum,1):
    plt.subplot(2,2,i)
    sns.countplot(data=dfa,y=col)
    plt.subplot(2,2,i+2)
    salary[col].value_counts(normalize=True).plot.bar()
    plt.ylabel(col)
    plt.xlabel('% distribution per category')
plt.tight_layout()
plt.show()
```

<Figure size 1152x720 with 0 Axes>

In [30]:
```
num_columns = dfa.select_dtypes(exclude='object').columns.tolist()
```

In [32]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [47]:
```
plt.figure(figsize=(18,40))
for i,col in enumerate(num_columns,1):
    plt.subplot(8,4,i)
    sns.kdeplot(salary[col],color='g',shade=True)
    plt.subplot(8,4,i+10)
    salary[col].plot.box()
plt.tight_layout()
plt.show()
num_data = salary[num_columns]
pd.DataFrame(data=[num_data.skew(),num_data.kurtosis()],index=['skewnes
```



Out[47]:

	age	workclass	education	educationno	maritalstatus	occupation	relatio
skewness	0.532784	1.148931	-0.945666	-0.310621	-0.006760	0.107141	0.8
kurtosis	-0.155931	2.329983	0.773506	0.635045	-0.538981	-1.249883	-0.6

SVM

In [48]:
```
col = salary.columns
```

In [49]:
```
x_train = train[col[0:13]]
y_train = train[col[13]]
x_test = test[col[0:13]]
y_test = test[col[13]]
```

In [50]:
```
def norm_func(i):
    x = (i-i.min())/(i.max()-i.min())
    return (x)
```

In [51]:
```
x_train = norm_func(x_train)
x_test =  norm_func(x_test)
```

Linear

In [56]:
```
model_linear = SVC(kernel = "linear")
model_linear.fit(x_train,y_train)
pred_test_linear = model_linear.predict(x_test)
print("Accuracy:",metrics.accuracy_score(y_test, pred_test_linear))
```

```
Accuracy: 0.8097609561752988
```

Poly

In [59]:
```
model_poly = SVC(kernel = "poly")
model_poly.fit(x_train,y_train)
pred_test_poly = model_poly.predict(x_test)
print("Accuracy:",metrics.accuracy_score(y_test, pred_test_poly))
```

```
Accuracy: 0.8435590969455511
```

RBF

In [60]:
```
model_rbf = SVC(kernel = "rbf")
model_rbf.fit(x_train,y_train)
pred_test_rbf = model_rbf.predict(x_test)
print("Accuracy:",metrics.accuracy_score(y_test, pred_test_rbf))
```

```
Accuracy: 0.8432934926958832
```

Sigmoid

In [61]:
```
model_sigmoid = SVC(kernel = "sigmoid")
model_sigmoid.fit(x_train,y_train)
pred_test_sigmoid = model_sigmoid.predict(x_test)
print("Accuracy:",metrics.accuracy_score(y_test, pred_test_sigmoid))
```

```
Accuracy: 0.5768924302788845
```

PolyModel gives the best Accuracy

In [ ]: