1. Import necessery libraries

In [1]:

```
import numpy as np
import pandas as pd
from sklearn import preprocessing
from sklearn import metrics
import seaborn as sns
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import LabelEncoder
from sklearn import preprocessing
from mlxtend.plotting import plot_decision_regions
```

Problem

Prepare a classification model using SVM for salary data

2. Import data

In [2]:

```
test_data = pd.read_csv('SalaryData_Test(1).csv')
train_data = pd.read_csv('SalaryData_Train(1).csv')
```

In [3]:

```
test = test_data.copy()
train = train_data.copy()
```

In [4]:

```
test.head()
```

Out[4]:

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

In [5]:

```
train.head()
```

Out[5]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex
0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex
1	50	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male
2	38	Private	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male
3	53	Private	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male

Data understanding

In [6]:

```
train.shape,test.shape
```

Out[6]:

```
((30161, 14), (15060, 14))
```

In [7]:

```
train.isna().sum(),test.isna().sum()
```

Out[7]:

```
(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,
 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 [8]:

```
train.dtypes,test.dtypes
```

Out[8]:

```
(age               int64
 workclass        object
 education        object
 educationno       int64
 maritalstatus    object
 occupation       object
 relationship     object
```

```
race              object
sex               object
capitalgain        int64
capitalloss        int64
hoursperweek       int64
native            object
Salary            object
dtype: object,
age                int64
workclass         object
education         object
educationno        int64
maritalstatus     object
occupation        object
relationship      object
race              object
sex               object
capitalgain        int64
capitalloss        int64
hoursperweek       int64
native            object
Salary            object
dtype: object)
```

In [9]:
```
str_c = ["workclass","education","maritalstatus","occupation","relationshi
```

In [10]:
```
model = LabelEncoder()
```

In [11]:
```
for i in str_c:
    train[i]= model.fit_transform(train[i])
    test[i]=model.fit_transform(test[i])
```

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

Out[12]:

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

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

Out[13]:

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

In [14]:
```
mapping = {' >50K': 1, ' <=50K': 2}
```

In [15]:
```
train = train.replace({'Salary': mapping})
test = test.replace({'Salary': mapping})
```

In [16]:
```
df = train.append(test)
```

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

In [18]:
```
salary_data.head()
```

Out[18]:

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

In [19]:
```
salary_data.shape
```

Out[19]:
```
(45221, 14)
```

In [20]:
```
salary_data.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

	count	mean	std	min	25%	50%	75%	max

In [21]:
```
salary_data.isnull().sum()
```

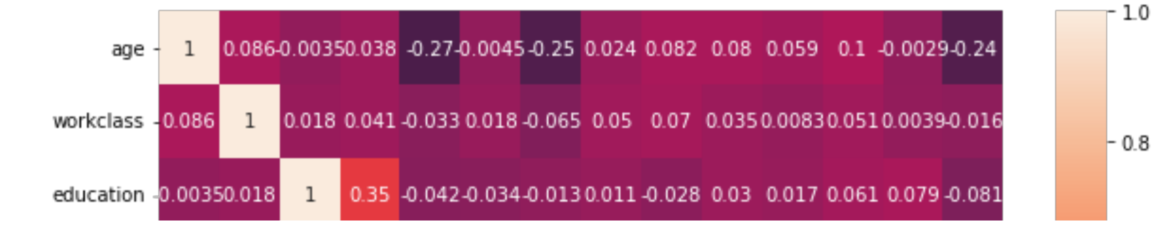
Out[21]:
```
age              0
workclass        0
education        0
educationno      0
maritalstatus    0
occupation       0
relationship     0
race             0
sex              0
capitalgain      0
capitalloss      0
hoursperweek     0
native           0
Salary           0
dtype: int64
```

3.Finding Correlation

In [22]:
```
corr = salary_data.corr()
```

In [23]:
```
plt.figure(figsize=(10,10))
sns.heatmap(corr,annot=True)
```

Out[23]:
```
<AxesSubplot:>
```



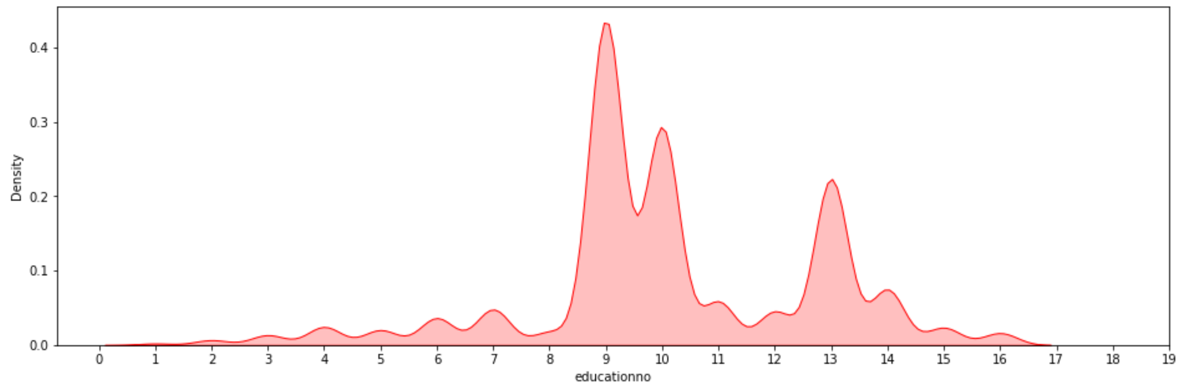
In [24]:

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

In [29]:

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

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



The Data is negatively skewed and has low kurtosis value

Most of people have eduction Number of years of education 9 - 10

In [26]:

```
salary_data.reset_index(inplace = True)
```

In [27]:

```
dfa = salary_data[salary_data.columns[0:13]]
obj_colum = dfa.select_dtypes(include='object').columns.tolist()
```

In [28]:

```
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_data[col].value_counts(normalize=True).plot.bar()
    plt.ylabel(col)
    plt.xlabel('% distribution per category')
plt.tight_layout()
plt.show()
```

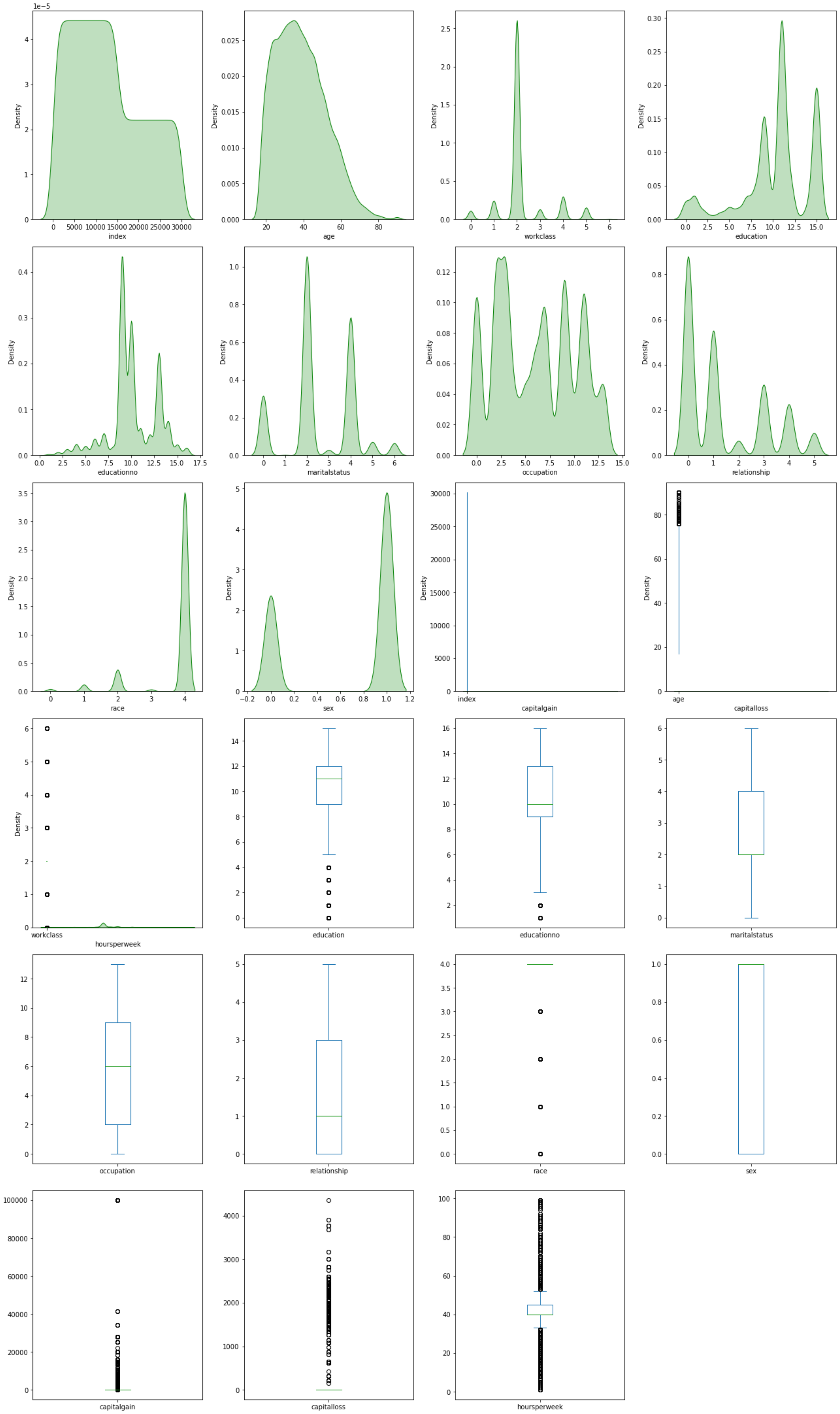
```
<Figure size 1152x720 with 0 Axes>
```

Majority of the workclass is in Private Sector

Also for education mojority of the people have HS graduation or some college degree

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

In [31]:
```
plt.figure(figsize=(18,40))
for i,col in enumerate(num_columns,1):
    plt.subplot(8,4,i)
    sns.kdeplot(salary_data[col],color='g',shade=True)
    plt.subplot(8,4,i+10)
    salary_data[col].plot.box()
plt.tight_layout()
plt.show()
num_data = salary_data[num_columns]
pd.DataFrame(data=[num_data.skew(),num_data.kurtosis()],index=['skewness',
```



Out[31]:

	index	age	workclass	education	educationno	maritalstatus	occupation	rel

	index	age	workclass	education	educationno	maritalstatus	occupation	rel
skewness	0.438900	0.532784	1.148931	-0.945666	-0.310621	-0.006760	0.107141	

4. SVM

In [32]:
```
col = salary_data.columns
```

In [33]:
```
x = salary_data.iloc[:,0:13]
y = salary_data.iloc[:,13]
```

In [34]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.20, stra
```

4.1 Linear

In [ ]:
```
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))
```

4.2 poly

In [ ]:
```
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))
```

4.3 RBF

In [ ]:
```
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))
```

4.4 Sigmoid

In [ ]:
```
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))
```

5. conclusion

Poly Model gives the best accuracy

In [ ]:

```

```