# SVM (Salary_Data)

1) 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

# 1. Import Libs

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 mlxtend.plotting import plot_decision_regions
from sklearn.metrics import confusion_matrix as cm
from sklearn.metrics import accuracy_score as ac
from sklearn.metrics import classification_report as report,roc_curve
from sklearn.metrics import roc_auc_score
from sklearn.preprocessing import LabelEncoder
```

# 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_data
```

Out[3]:

| | 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 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **15055** | 33 | Private | Bachelors | 13 | Never-married | Prof-specialty | Own-child | White |
| **15056** | 39 | Private | Bachelors | 13 | Divorced | Prof-specialty | Not-in-family | White |
| **15057** | 38 | Private | Bachelors | 13 | Married-civ-spouse | Prof-specialty | Husband | White |
| **15058** | 44 | Private | Bachelors | 13 | Divorced | Adm-clerical | Own-child | Asian-Pac-Islander |
| **15059** | 35 | Self-emp-inc | Bachelors | 13 | Married-civ-spouse | Exec-managerial | Husband | White |

15060 rows × 14 columns

In [4]:

```
train_data
```

Out[4]:

| | 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 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **30156** | 27 | Private | Assoc-acdm | 12 | Married-civ-spouse | Tech-support | Wife | White |
| **30157** | 40 | Private | HS-grad | 9 | Married-civ-spouse | Machine-op-inspct | Husband | White |
| **30158** | 58 | Private | HS-grad | 9 | Widowed | Adm-clerical | Unmarried | White |
| **30159** | 22 | Private | HS-grad | 9 | Never-married | Adm-clerical | Own-child | White |
| **30160** | 52 | Self-emp-inc | HS-grad | 9 | Married-civ-spouse | Exec-managerial | Wife | White |

30161 rows × 14 columns

# 3. Data Preprocessing

In [5]:

```
df_temp = test_data.append(train_data)
df_temp
```

```
C:\Users\shubham\AppData\Local\Temp\ipykernel_2088\1463285300.py:1: FutureWa
rning: The frame.append method is deprecated and will be removed from pandas
in a future version. Use pandas.concat instead.
  df_temp = test_data.append(train_data)
```

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 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **30156** | 27 | Private | Assoc-acdm | 12 | Married-civ-spouse | Tech-support | Wife | White |
| **30157** | 40 | Private | HS-grad | 9 | Married-civ-spouse | Machine-op-inspct | Husband | White |
| **30158** | 58 | Private | HS-grad | 9 | Widowed | Adm-clerical | Unmarried | White |
| **30159** | 22 | Private | HS-grad | 9 | Never-married | Adm-clerical | Own-child | White |
| **30160** | 52 | Self-emp-inc | HS-grad | 9 | Married-civ-spouse | Exec-managerial | Wife | White |

45221 rows × 14 columns

In [6]:

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

Out[6]:

| | 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 [7]:

```
train.head()
```

Out[7]:

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

In [8]:

```
str_c = ["workclass","education","maritalstatus","occupation","relationship","race","sex","
str_c
```

Out[8]:

```
['workclass',
 'education',
 'maritalstatus',
 'occupation',
 'relationship',
 'race',
 'sex',
 'native']
```

In [9]:

```
for i in str_c:
    train[i]= LabelEncoder().fit_transform(train[i])
    test[i]=LabelEncoder().fit_transform(test[i])

mapping = {' >50K': 1, ' <=50K': 0}
train = train.replace({'Salary': mapping})
test = test.replace({'Salary': mapping})
```

In [10]:

```
test.head()
```

Out[10]:

| | age | workclass | education | educationno | maritalstatus | occupation | relationship | race | sex |
|---|---|---|---|---|---|---|---|---|---|
| **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 [11]:

```
train.head()
```

Out[11]:

| | age | workclass | education | educationno | maritalstatus | occupation | relationship | race | sex |
|---|---|---|---|---|---|---|---|---|---|
| **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 [12]:

```
df = train.append(test)
```

```
C:\Users\shubham\AppData\Local\Temp\ipykernel_2088\2767323617.py:1: FutureWa
rning: The frame.append method is deprecated and will be removed from pandas
in a future version. Use pandas.concat instead.
  df = train.append(test)
```

In [13]:

```
df1 = df.copy()
df1
```

Out[13]:

| | age | workclass | education | educationno | maritalstatus | occupation | relationship | race | s |
|---|---|---|---|---|---|---|---|---|---|
| **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 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **15055** | 33 | 2 | 9 | 13 | 4 | 9 | 3 | 4 | |
| **15056** | 39 | 2 | 9 | 13 | 0 | 9 | 1 | 4 | |
| **15057** | 38 | 2 | 9 | 13 | 2 | 9 | 0 | 4 | |
| **15058** | 44 | 2 | 9 | 13 | 0 | 0 | 3 | 1 | |
| **15059** | 35 | 3 | 9 | 13 | 2 | 3 | 0 | 4 | |

45221 rows × 14 columns

In [14]:

```
df1.describe()
```

Out[14]:

| | age | workclass | education | educationno | maritalstatus | occupation | r |
|---|---|---|---|---|---|---|---|
| **count** | 45221.000000 | 45221.000000 | 45221.000000 | 45221.000000 | 45221.000000 | 45221.000000 | 45 |
| **mean** | 38.548086 | 2.204507 | 10.313217 | 10.118463 | 2.585148 | 5.969572 | |
| **std** | 13.217981 | 0.958132 | 3.816992 | 2.552909 | 1.500460 | 4.026444 | |
| **min** | 17.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 | |
| **25%** | 28.000000 | 2.000000 | 9.000000 | 9.000000 | 2.000000 | 2.000000 | |
| **50%** | 37.000000 | 2.000000 | 11.000000 | 10.000000 | 2.000000 | 6.000000 | |
| **75%** | 47.000000 | 2.000000 | 12.000000 | 13.000000 | 4.000000 | 9.000000 | |
| **max** | 90.000000 | 6.000000 | 15.000000 | 16.000000 | 6.000000 | 13.000000 | |

In [15]:

```
df1.isna().sum()
```

Out[15]:

```
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
```

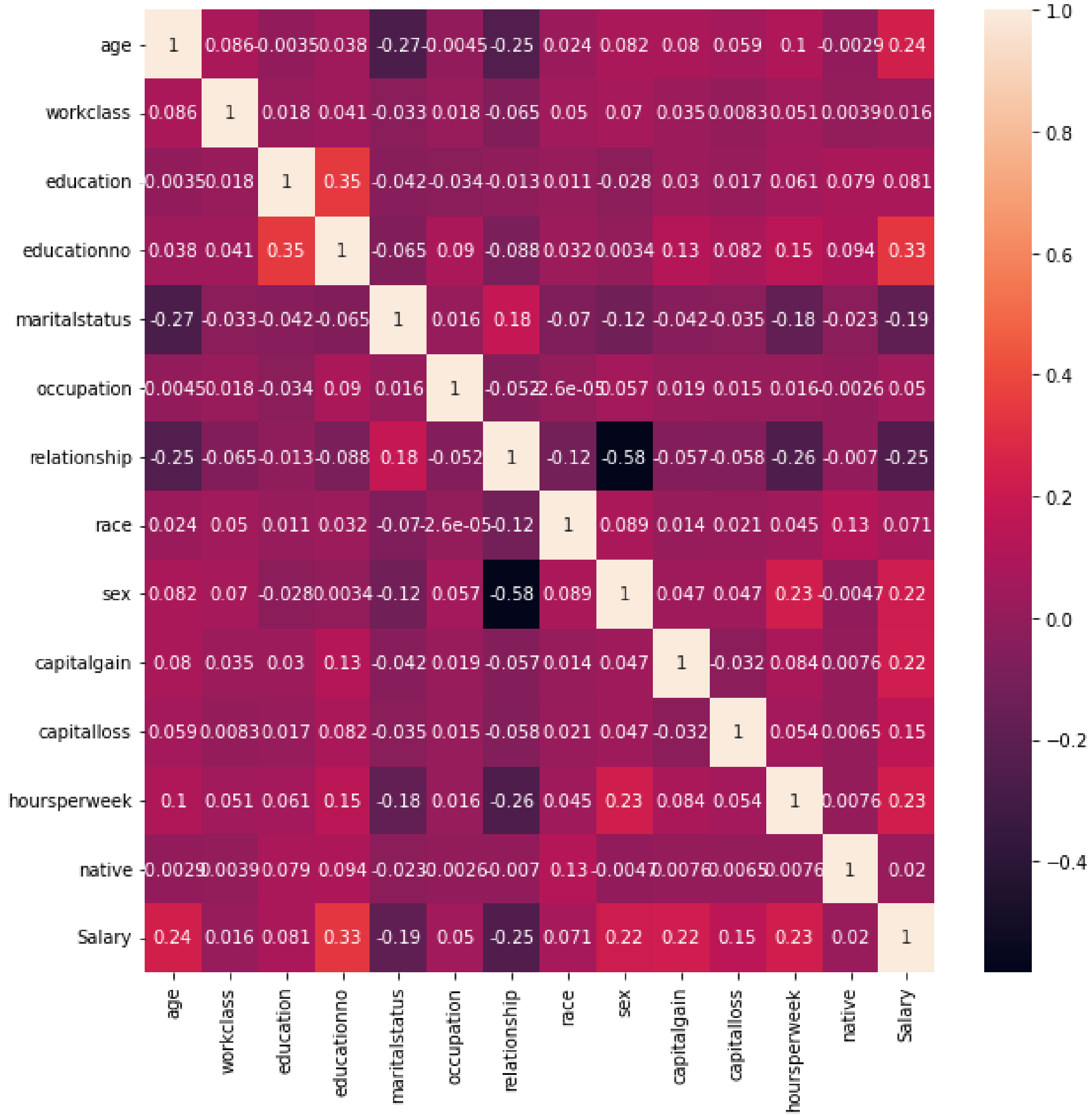
## Finding Correlation

In [16]:

```
corr = df1.corr()
```

In [17]:

```
plt.figure(figsize=(10,10))
sns.heatmap(corr,annot=True)
plt.show()
```



In [18]:

```
df1.index.is_unique
```

Out[18]:

```
False
```

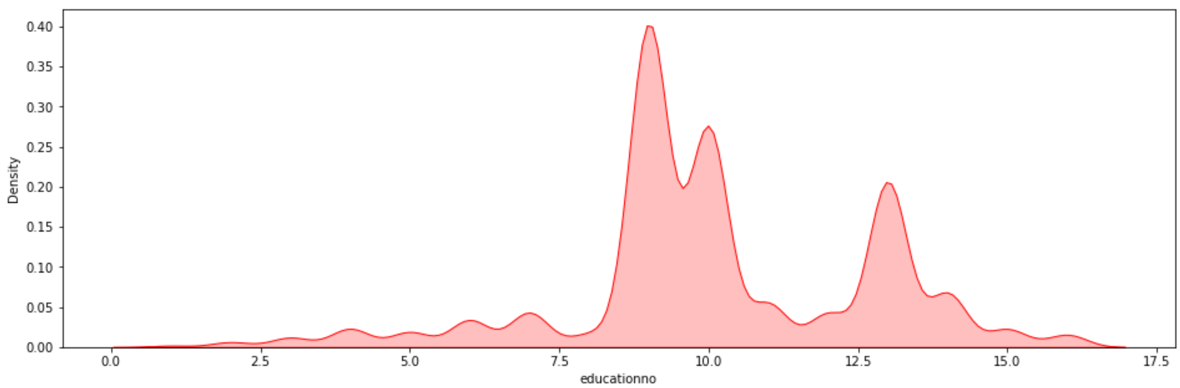
In [19]:

```
df1=df1.loc[~df1.index.duplicated(), :]
```

In [20]:

```
plt.figure(figsize=(16,5))
print("Skew: {}".format(df1['educationno'].skew()))
print("Kurtosis: {}".format(df1['educationno'].kurtosis()))
sns.kdeplot(df1['educationno'],shade=True,color='r')
plt.show()
```

```
Skew: -0.305378355820322
Kurtosis: 0.643604835875955
```




**The Data is negatively skewed and has Low Kurtosis value**

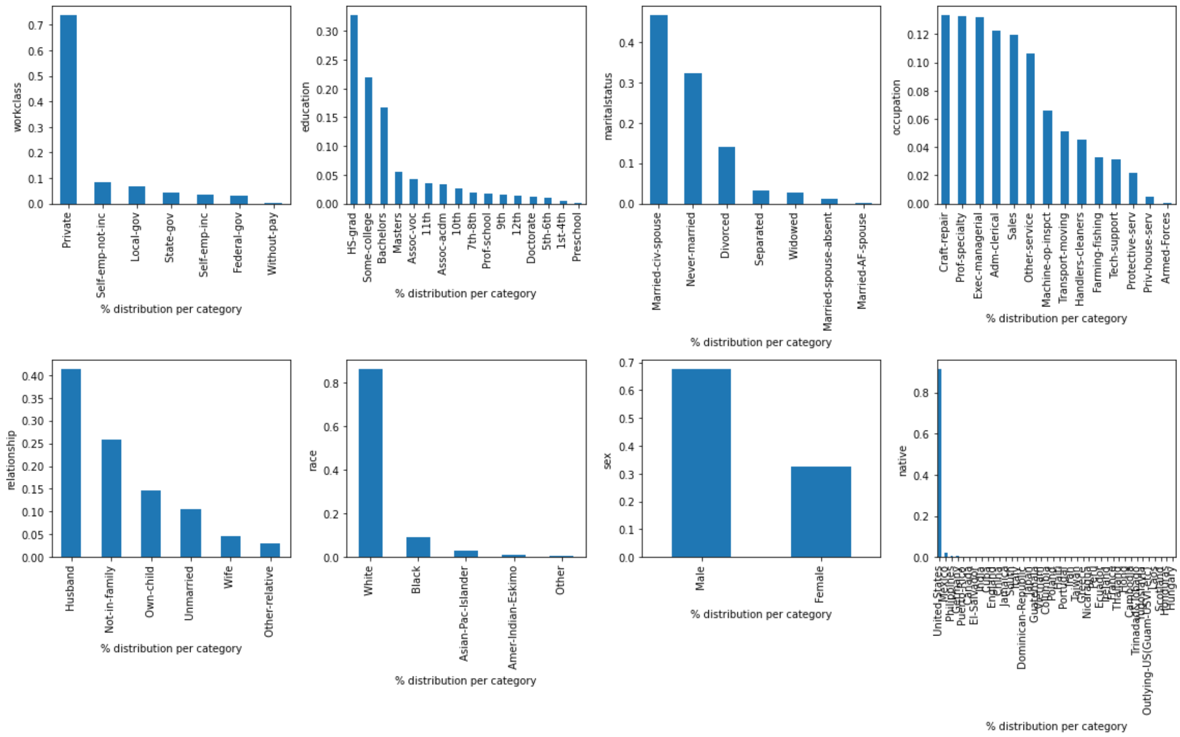
**Most of people have eduction Number of years of education 8 - 11**

In [21]:

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

In [22]:

```
plt.figure(figsize=(16,10))
for i,col in enumerate(obj_colum,1):
    plt.subplot(2,4,i)
    df_temp[col].value_counts(normalize=True).plot.bar()
    plt.ylabel(col)
    plt.xlabel('% distribution per category')
plt.tight_layout()
plt.show()
```

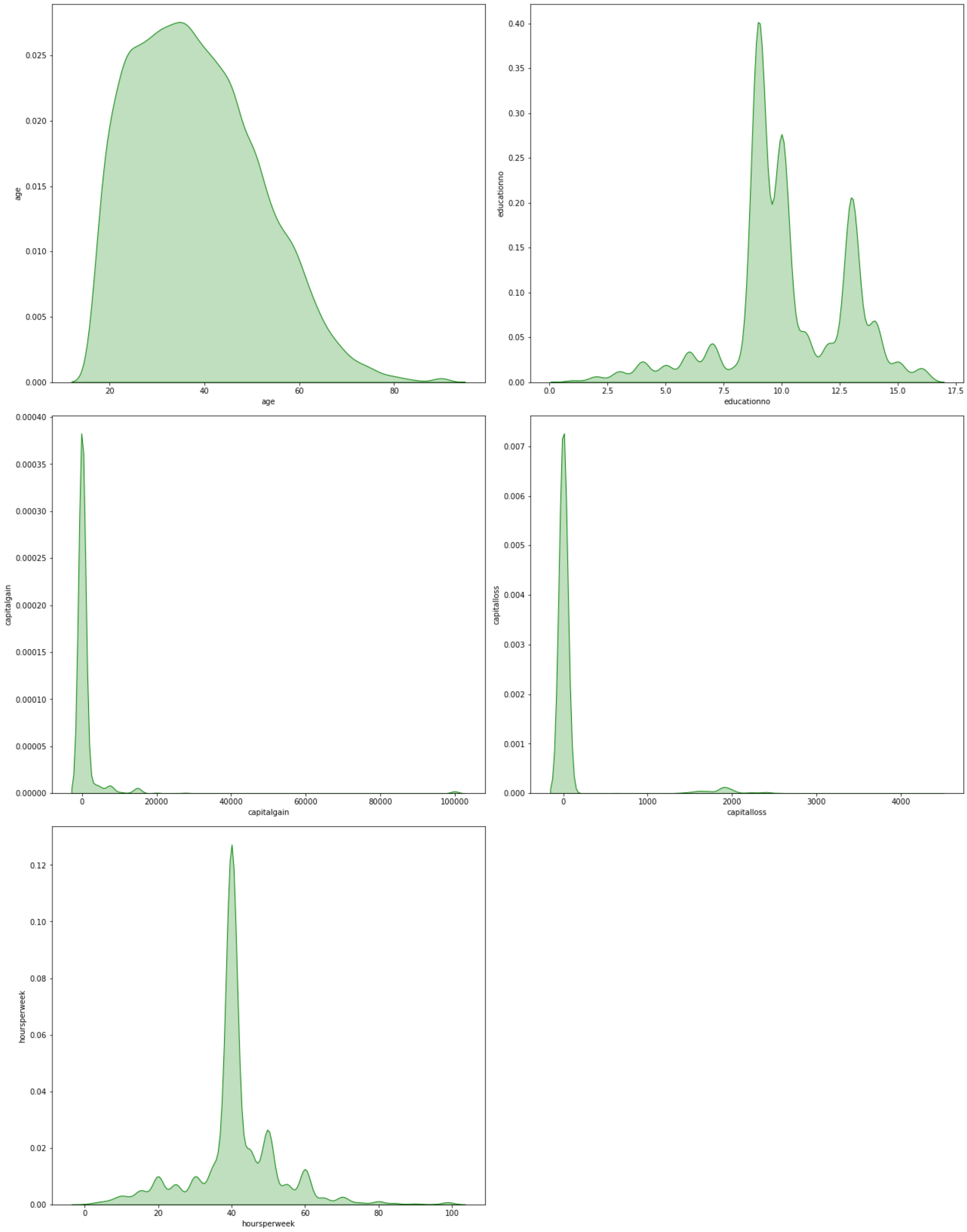



In [23]:

```
num_columns = dfa.select_dtypes(exclude='object')
```

In [24]:

```
plt.figure(figsize=(18,30))
for i,col in enumerate(num_columns,1):
    plt.subplot(4,2,i)
    sns.kdeplot(df1[col],color='g',shade=True,legend=True)
    plt.ylabel(col)
plt.tight_layout()
plt.show()
```



In [25]:

```
pd.DataFrame(data=[num_columns.skew(),num_columns.kurtosis()],index=['skewness','kurtosis']
```

Out[25]:

| | age | educationno | capitalgain | capitalloss | hoursperweek |
|---|---|---|---|---|---|
| **skewness** | 0.532784 | -0.310621 | 11.788871 | 4.517536 | 0.340536 |
| **kurtosis** | -0.155931 | 0.635045 | 150.147899 | 19.376085 | 3.201287 |

# 4. Model Building

## SVM

In [26]:

```
col = df1.columns
col
```

Out[26]:

```
Index(['age', 'workclass', 'education', 'educationno', 'maritalstatus',
       'occupation', 'relationship', 'race', 'sex', 'capitalgain',
       'capitalloss', 'hoursperweek', 'native', 'Salary'],
      dtype='object')
```

In [69]:

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

In [70]:

```
def norm_func(i):
    x = (i-i.min())/(i.max()-i.min())
    return (x)
```

In [71]:

```
x_train = norm_func(x_train)
x_test =  norm_func(x_test)
```

# 4.1 Linear

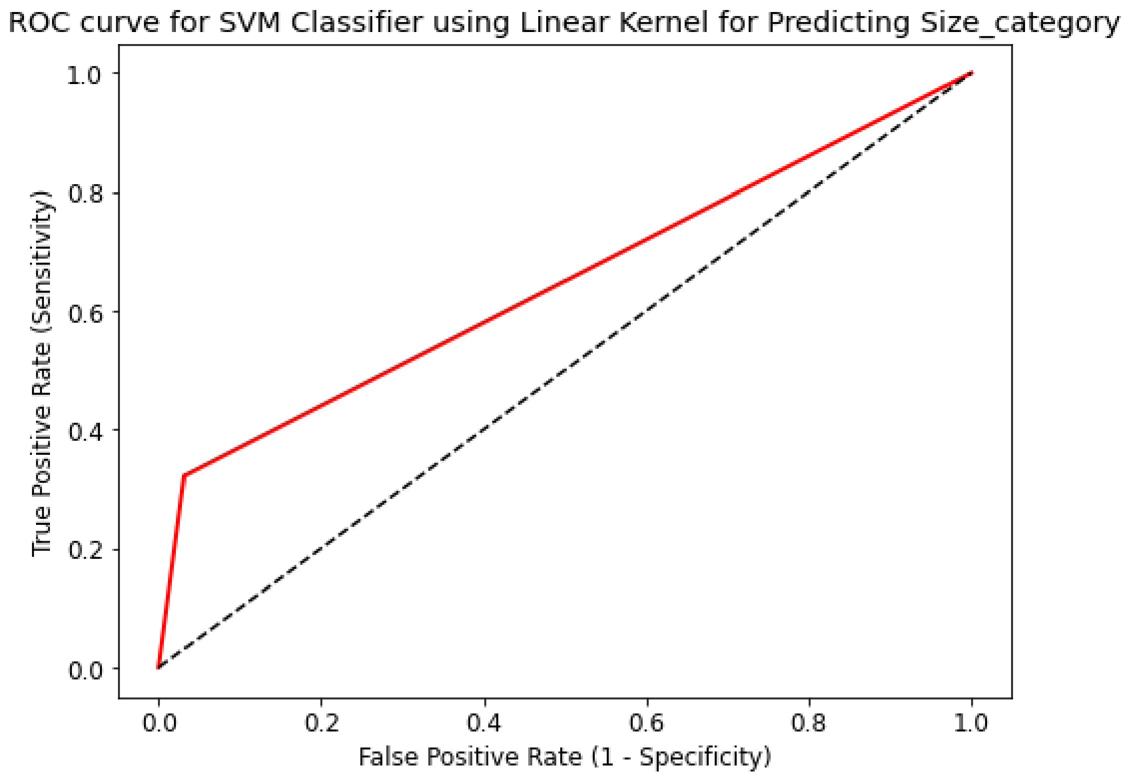
In [72]:

```
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: 0.8098273572377158
```

In [73]:

```
fpr, tpr, thresholds = roc_curve(y_test, pred_test_linear)
plt.figure(figsize=(8,6))
plt.plot(fpr, tpr, linewidth=2, color='red')
plt.plot([0,1], [0,1], 'k--' )
plt.rcParams['font.size'] = 12
plt.title('ROC curve for SVM Classifier using Linear Kernel for Predicting Size_category')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.show()
ROC_AUC = roc_auc_score(y_test, pred_test_linear)
print('ROC AUC : {:.4f}'.format(ROC_AUC))
```




```
ROC AUC : 0.6455
```

# 4.2 Poly

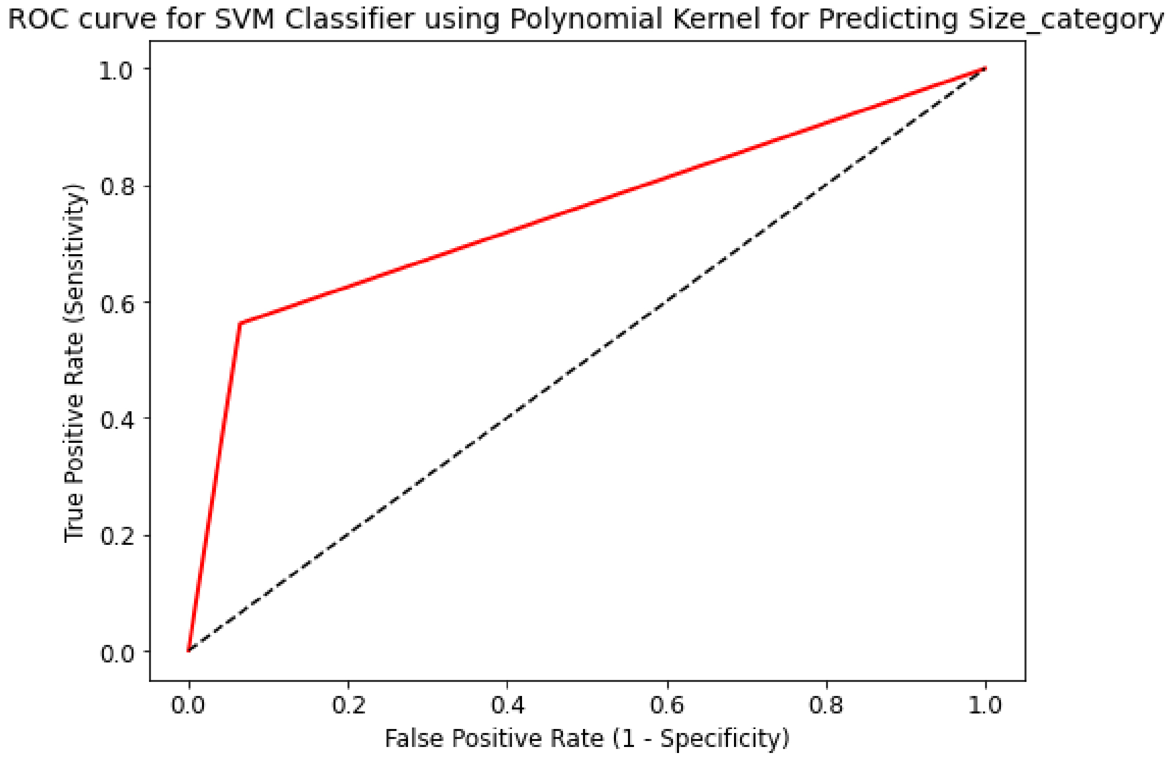
In [74]:

```
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
```

In [75]:

```
fpr, tpr, thresholds = roc_curve(y_test, pred_test_poly)
plt.figure(figsize=(8,6))
plt.plot(fpr, tpr, linewidth=2, color='red')
plt.plot([0,1], [0,1], 'k--' )
plt.rcParams['font.size'] = 12
plt.title('ROC curve for SVM Classifier using Polynomial Kernel for Predicting Size_categor
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.show()
ROC_AUC = roc_auc_score(y_test, pred_test_poly)
print('ROC AUC : {:.4f}'.format(ROC_AUC))
```




```
ROC AUC : 0.7485
```

# 4.3 RBF

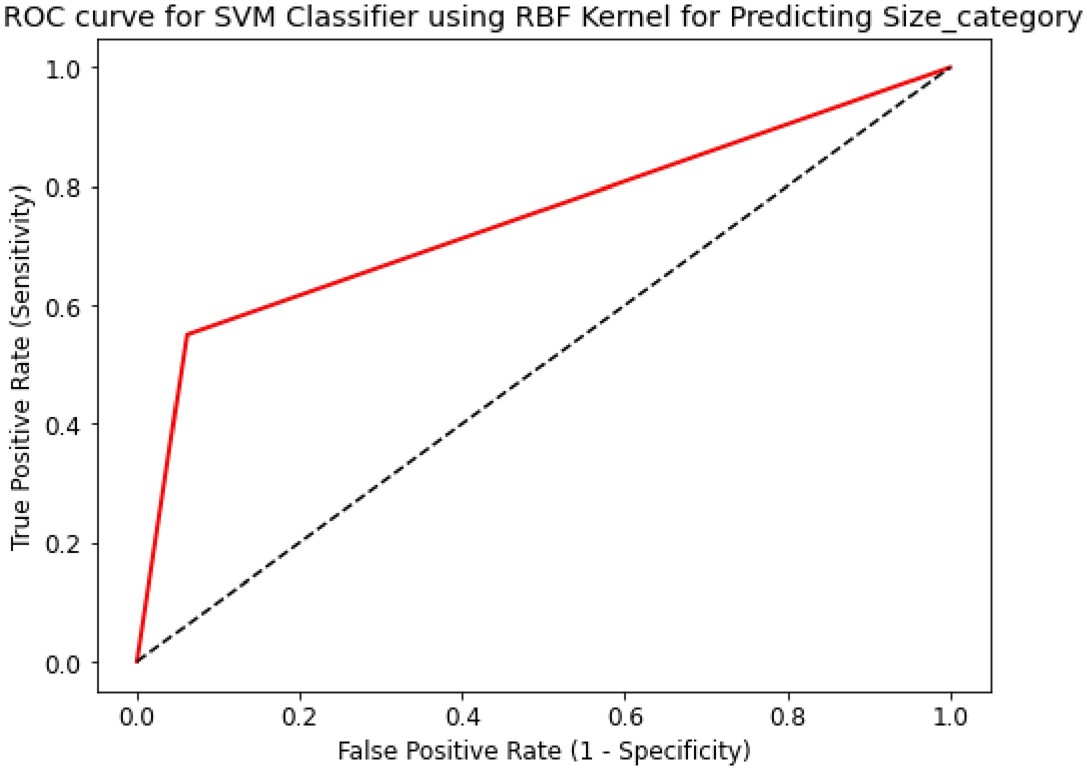
In [76]:

```
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
```

In [77]:

```
fpr, tpr, thresholds = roc_curve(y_test, pred_test_rbf)
plt.figure(figsize=(8,6))
plt.plot(fpr, tpr, linewidth=2, color='red')
plt.plot([0,1], [0,1], 'k--' )
plt.rcParams['font.size'] = 12
plt.title('ROC curve for SVM Classifier using RBF Kernel for Predicting Size_category')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.show()
ROC_AUC = roc_auc_score(y_test, pred_test_rbf)
print('ROC AUC : {:.4f}'.format(ROC_AUC))
```




```
ROC AUC : 0.7445
```

## 4.4 Sigmoid

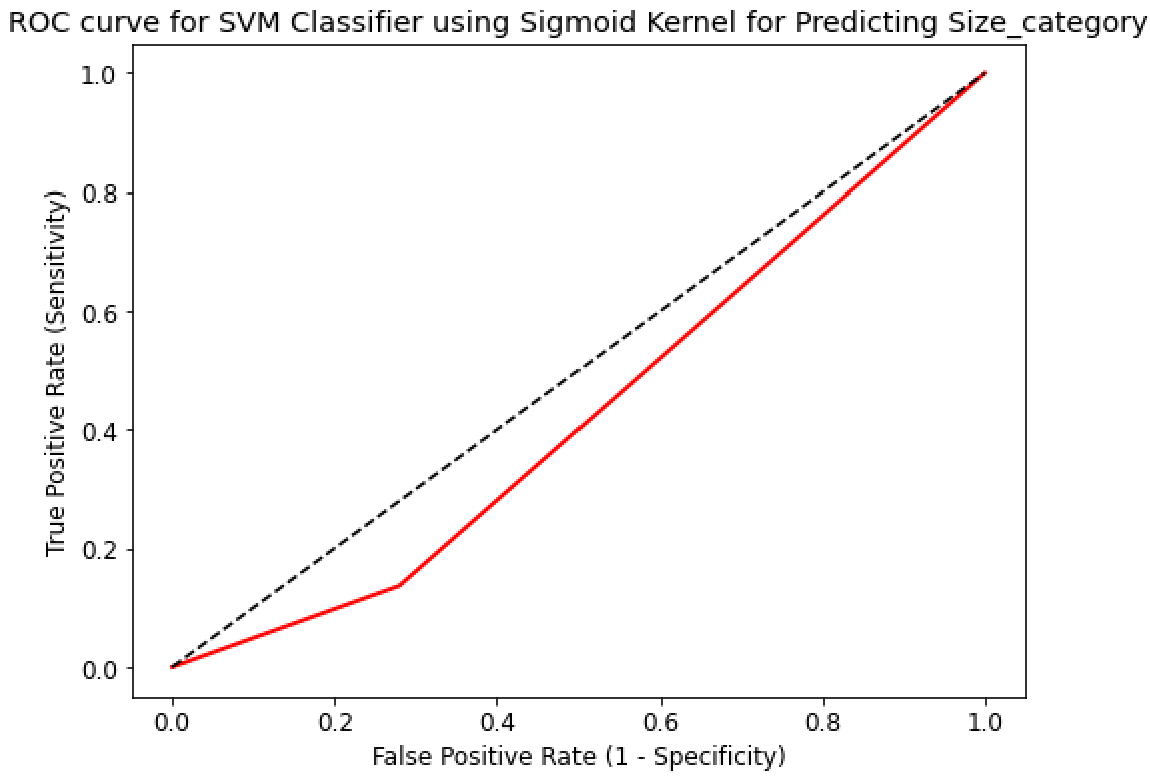
In [78]:

```
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
```

In [79]:

```
fpr, tpr, thresholds = roc_curve(y_test, pred_test_sigmoid)
plt.figure(figsize=(8,6))
plt.plot(fpr, tpr, linewidth=2, color='red')
plt.plot([0,1], [0,1], 'k--' )
plt.rcParams['font.size'] = 12
plt.title('ROC curve for SVM Classifier using Sigmoid Kernel for Predicting Size_category')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.show()
ROC_AUC = roc_auc_score(y_test, pred_test_sigmoid)
print('ROC AUC : {:.4f}'.format(ROC_AUC))
```




```
ROC AUC : 0.4285
```

## The Poly Model has best accuracy compare to other Models. but RBF model has almost equal Accuracy to Poly Model

In [ ]: