Importing Necessary Liabrary

In [67]:

```
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
from sklearn.preprocessing import LabelEncoder
from mlxtend.plotting import plot_decision_regions
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
from matplotlib.colors import ListedColormap
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

Business problem

Prepare a model on salary data using Naive bayes algorithm

Data collection and description

Importing dataset

In [14]:

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

In [15]:

```
train_data.head()
```

Out[15]:

	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 [16]:

```
test_data.head()
```

Out[16]:

	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 [17]:

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

In [18]:

```
encoder = LabelEncoder()
```

In [19]:

```
for i in category:
    train_data[i]= encoder.fit_transform(train_data[i])
    test_data[i]=encoder.fit_transform(test_data[i])
```

In [20]:

```
train_data.head()
```

Out[20]:

	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 [21]:

```
mapping = {' >50K': 1, ' <=50K': 0}
```

In [22]:

```
train_data = train_data.replace({'Salary': mapping})
test_data = test_data.replace({'Salary': mapping})
```

In [23]:

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

In [24]:

```
salary_df.head()
```

Out[24]:

	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

Initial Analysis

In [25]:

```
salary_df.dtypes
```

Out[25]:

```
age              int64
workclass        int32
education        int32
educationno      int64
maritalstatus    int32
occupation       int32
relationship     int32
race             int32
sex              int32
capitalgain      int64
capitalloss      int64
hoursperweek     int64
native           int32
Salary           int64
dtype: object
```

In [26]:

```
salary_df.isna().sum()
```

Out[26]:

```
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 [31]:

```
salary_df.describe().T
```

Out[31]:

	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	0.247849	0.431769	0.0	0.0	0.0	0.0	1.0

In [27]:

```
salary_df.shape
```

Out[27]:

(45221, 14)

Finding correlation

In [35]:

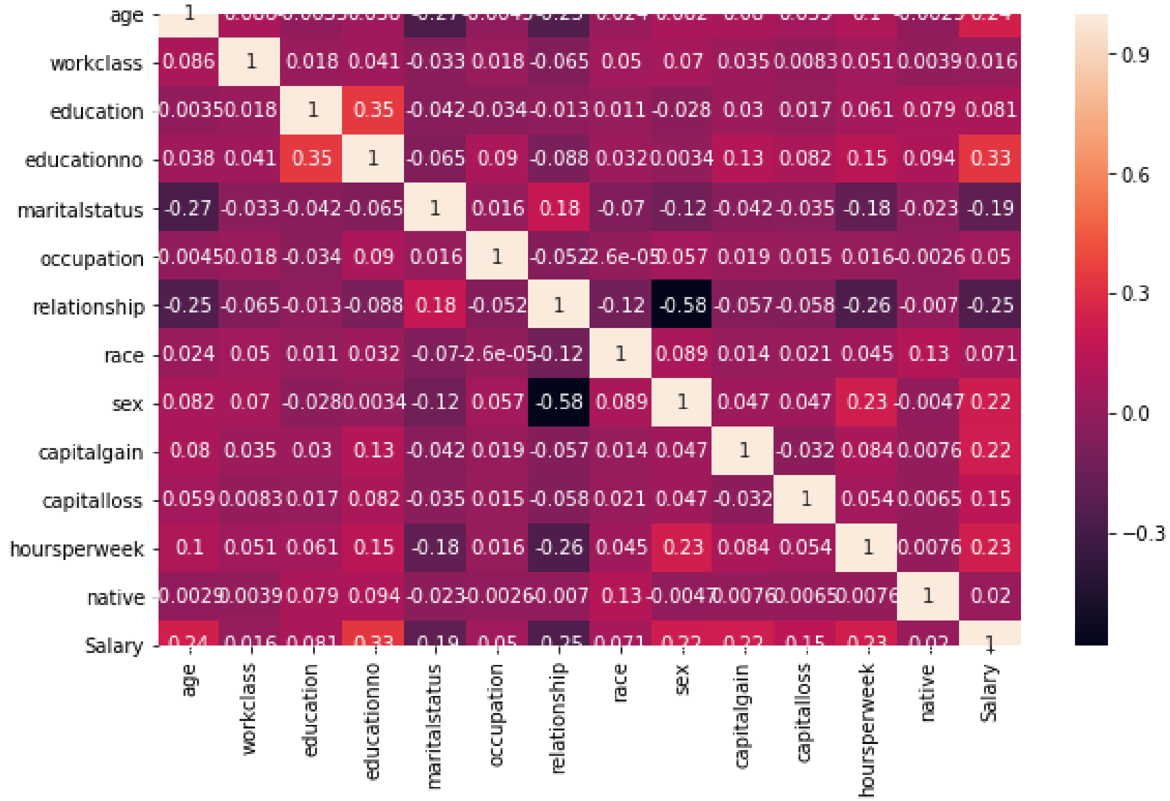
```
corr = salary_df.corr()
```

In [39]:

```
plt.figure(figsize=(10,6))
sns.heatmap(corr,annot=True)
```

Out[39]:

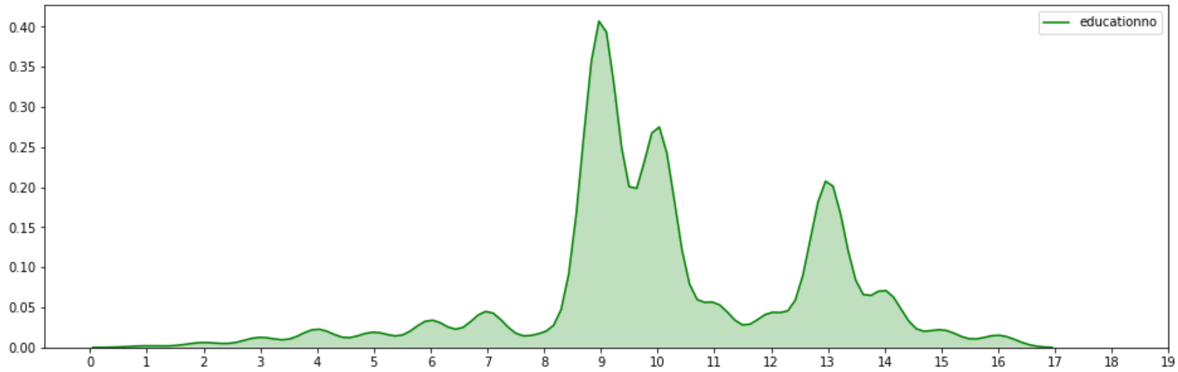
```
<matplotlib.axes._subplots.AxesSubplot at 0x22149f87f48>
```



In [40]:

```
plt.figure(figsize=(16,5))
print("Skew: {}".format(salary_df['educationno'].skew()))
print("Kurtosis: {}".format(salary_df['educationno'].kurtosis()))
ax = sns.kdeplot(salary_df['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 negatively skewed and has low kurtosis value

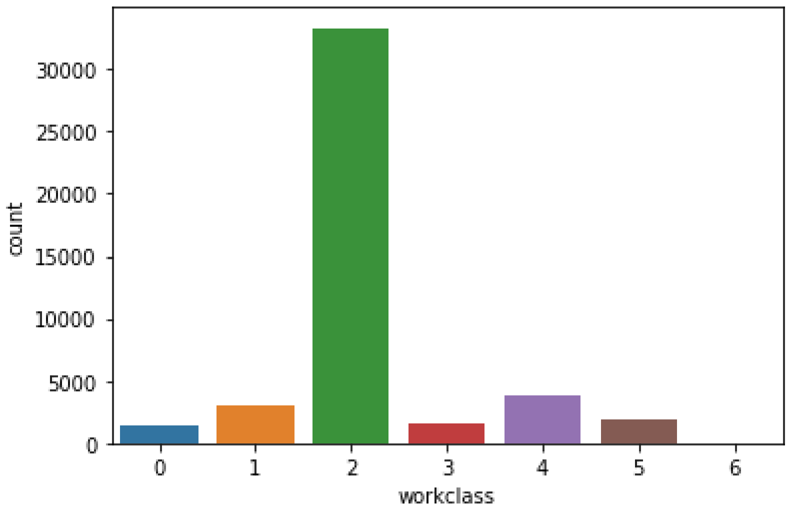
Most of the people have educationno is 8-11

In [45]:

```
sns.countplot(x='workclass',data=salary_df)
```

Out[45]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2214a52f708>
```

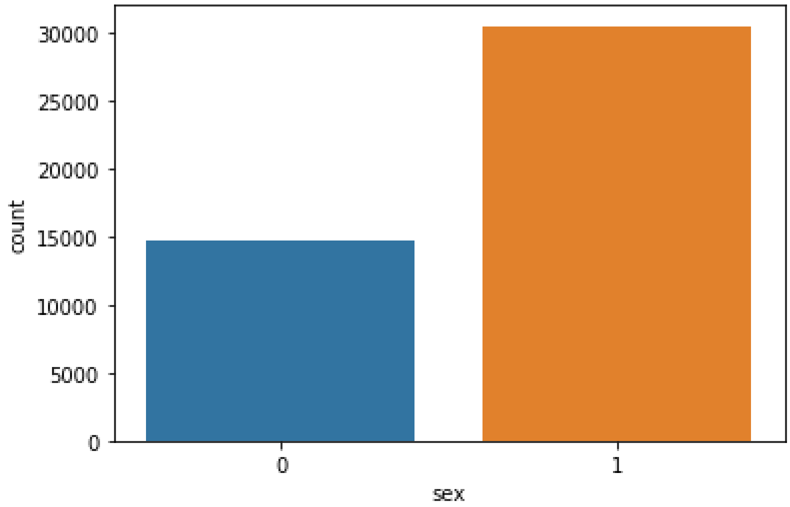


In [47]:

```
sns.countplot(x='sex',data=salary_df)
```

Out[47]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2214ad042c8>
```



The majority of workclass is Private sector denoted by 2

The majority of sex is Male.

Model Training| Model Testing| Model Evlauation

Naive Bayes

In [49]:

```
X_train = train_data.iloc[:,0:13]
y_train = train_data.iloc[:,13]
X_test = test_data.iloc[:,0:13]
y_test = test_data.iloc[:,13]
```

Gausian NB

In [50]:

```
clsfrgnb = GaussianNB()
clsfrgnb.fit(X_train,y_train)
```

Out[50]:

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

In [51]:

```
y_pred_test=clsfrgnb.predict(X_test)
```

In [52]:

```
print(confusion_matrix(y_test,y_pred_test))
```

```
[[10759   601]
 [ 2491  1209]]
```

In [53]:

```
pd.crosstab(y_test.values.flatten(),clsfrgnb)
```

Out[53]:

col_0	GaussianNB(priors=None, var_smoothing=1e-09)
row_0	
0	11360
1	3700

In [54]:

```
print ("Accuracy",np.mean(y_pred_test==y_test.values.flatten()))
```

```
Accuracy 0.7946879150066402
```

Multinomial NB

In [56]:

```
clsfrmnb1 = MultinomialNB()
```

In [57]:

```
clsfrmnb1.fit(X_train,y_train)
```

Out[57]:

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

In [60]:

```
y_pred_test1=clsfrmnb1.predict(X_test)
```

In [61]:

```
print(confusion_matrix(y_test,y_pred_test1))
```

```
[[10891   469]
 [ 2920   780]]
```

In [62]:

```
pd.crosstab(y_test.values.flatten(),clsfrmnb1)
```

Out[62]:

col_0 / row_0	MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
0	11360
1	3700

In [63]:

```
print ("Accuracy",np.mean(y_pred_test1==y_test.values.flatten()))
```

```
Accuracy 0.7749667994687915
```

In [76]:

```
pd.DataFrame([y_test,y_pred_test])
```

Out[76]:

	0	1	2	3	4	5	6	7	8	9	...	15050	15051	15052	15053	15054	15055	15056
Salary	0	0	1	1	0	1	0	0	1	0	...	0	0	0	0	0	0	0
Unnamed 0	0	0	0	1	0	1	0	0	1	0	...	0	0	0	0	0	0	0

2 rows × 15060 columns

Cross validation

Thus Gaussian NB has better accuracy hence we used Gaussian NB classifier

We also cross validate to know which algorithm is best suited for it.

In [71]:

```
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('RF',  RandomForestClassifier()))
models.append(('NB', GaussianNB()))
```

In [72]:

```
results = []
names = []
scoring = 'accuracy'
```

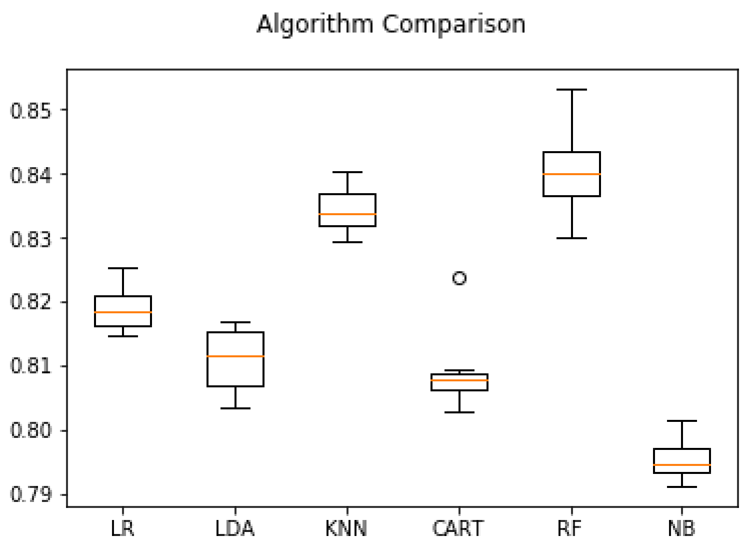
In [73]:

```
for name, model in models:
    kfold = model_selection.KFold(n_splits=7, random_state=21)
    cv_results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold, scoring
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
LR: 0.818806 (0.003710)
LDA: 0.810782 (0.005039)
KNN: 0.834422 (0.003710)
CART: 0.809191 (0.006279)
RF: 0.840456 (0.006901)
NB: 0.795431 (0.003456)
```

In [74]:

```
fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```



The above stated boxplot, we got the best RandomForest classification is best algorithm.

>>The End!! <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<