1. Import neccessry libraries

In [68]:

```
import numpy as np
import pandas as pd
from sklearn import preprocessing
from sklearn import metrics
import seaborn as sns
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
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
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 import svm
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
from sklearn.svm import SVC
from sklearn import model_selection

import warnings
warnings.filterwarnings('ignore')
```

Problem

Prepare a classification model using Naive Bayes for salary data

2. Import data

In [19]:

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

In [20]:

```
df_data_1 = test_data.append(train_data)
```

In [21]:

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

In [22]:

```
test.head()
```

Out[22]:

	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 [23]:

```
train.head()
```

Out[23]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex
0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male
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
4	28	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female

In [24]:

```
str_data = ["workclass","education","maritalstatus","occupation","relations
```

In [25]:

```
numbers = LabelEncoder()

for i in str_data:
        train[i]=numbers.fit_transform(train[i])
        test[i]=numbers.fit_transform(test[i])
```

In [26]:

```
train.head()
```

Out[26]:

	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 [27]:
```
test.head()
```

Out[27]:

	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 [28]:
```
mapping = {'>50K':1, '<=50K':2}
```

In [29]:
```
train = train.replace({'Salary': mapping})
test = test.replace({'Salary': mapping})
```

In [30]:
```
df_data_2 = train.append(test)
```

In [31]:
```
df_data_2.head()
```

Out[31]:

	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 [32]:
```
df_data_2.shape
```

Out[32]:
```
(45221, 14)
```

In [33]:
```
df_data_2.describe().T
```

Out[33]:

	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

	count	mean	std	min	25%	50%	75%	max
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

In [34]:

```
df_data_2.isnull().sum()
```

Out[34]:

```
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
```

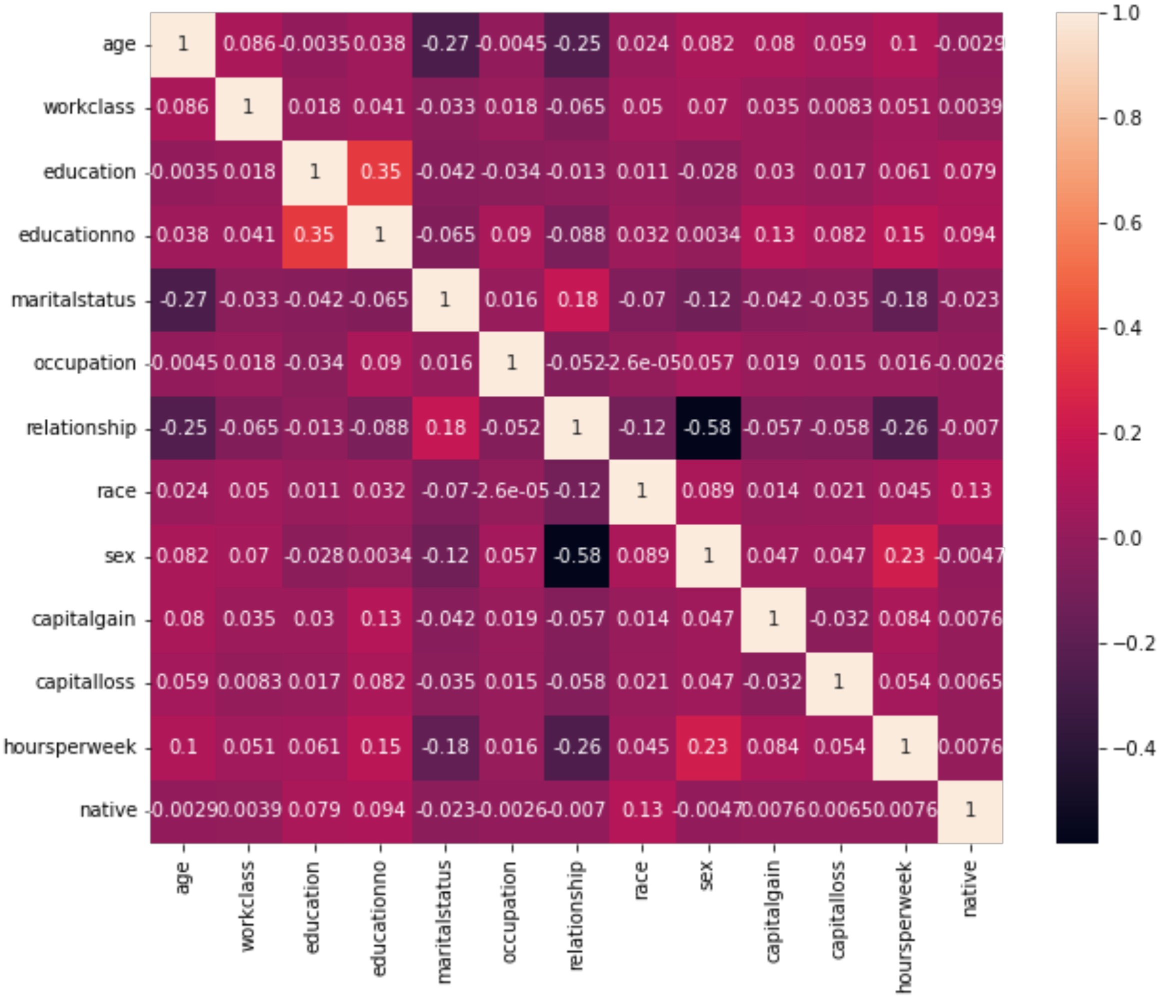
3.Finding correlation

In [35]:

```
corr = df_data_2.corr()
```

In [36]:

```
plt.figure(figsize=(10,8))
sns.heatmap(corr,annot=True)
plt.show()
```



In [37]:

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

In [49]:

```
plt.figure(figsize=(10,5))
print("Skew: {}".format(df_data_2['education'].skew()))
print("Kurtosis: {}".format(df_data_2['education'].kurtosis()))
ax = sns.kdeplot(df_data_2['education'],shade=True,color='b')
plt.xticks([i for i in range(0,20,1)])
plt.show()
```

```
Skew: -0.9456660524334967
Kurtosis: 0.7735061370983276
```

The Data is negatively skewed and has low kurtosis value

Most of people have eduction Number of years of education 10 - 13

In [46]:
```
df_data_2.reset_index(inplace = True)
```

In [39]:
```
df_data_2.index.duplicated()
```

Out[39]:
```
array([False, False, False, ...,  True,  True,  True])
```

In [40]:
```
df_data_2.index.is_unique
```

Out[40]:
```
False
```

In [41]:
```
idx = pd.Index(['race', 'sex', 'occupation', 'native', 'relationship'])
idx.duplicated()
```

Out[41]:
```
array([False, False, False, False, False])
```

4. Naive Bayes

In [50]:
```
x_train = train.iloc[:,0:13]
y_train = train.iloc[:,13]
x_test = test.iloc[:,0:13]
y_test = test.iloc[:,13]
```

4.1 GaussianNB

In [51]:
```
cls_gnb = GaussianNB()
```

In [52]:
```
cls_gnb.fit(x_train,y_train)
```

Out[52]:
```
GaussianNB()
```

In [53]:
```
y_pred_gnb = cls_gnb.predict(x_test)
```

In [54]:
```
confusion_matrix(y_test, y_pred_gnb)
```

Out[54]:
```
array([[10759,   601],
       [ 2491,  1209]], dtype=int64)
```

In [55]:
```
pd.crosstab(y_test.values.flatten(),cls_gnb)
```

Out[55]:

col_0	GaussianNB()
row_0	

col_0	GaussianNB()
row_0	

In [56]:
```
print ("Accuracy",np.mean(y_pred_gnb==y_test.values.flatten()))
```

```
Accuracy 0.7946879150066402
```

4.2 MultinomialNB

In [58]:
```
cls_mnb = MultinomialNB()
cls_mnb.fit(x_train,y_train)
```

Out[58]:
```
MultinomialNB()
```

In [59]:
```
y_pred_mnb = cls_mnb.predict(x_test)
```

In [60]:
```
confusion_matrix(y_test,y_pred_mnb)
```

Out[60]:
```
array([[10891,   469],
       [ 2920,   780]], dtype=int64)
```

In [61]:
```
pd.crosstab(y_test.values.flatten(),cls_mnb)
```

Out[61]:

col_0	MultinomialNB()
row_0	
<=50K	11360
>50K	3700

In [62]:
```
print ("Accuracy",np.mean(y_pred_mnb==y_test.values.flatten()))
```

```
Accuracy 0.7749667994687915
```

conclusion

GaussianNB Model has a better Accuracy, Thus we will use GaussianNB Classifier

We will also cross validate the model with other classifiers to get better understanding of which classifier is best suited for our data

In [63]:
```
seed = 7
```

In [64]:
```
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
```

In [65]:

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

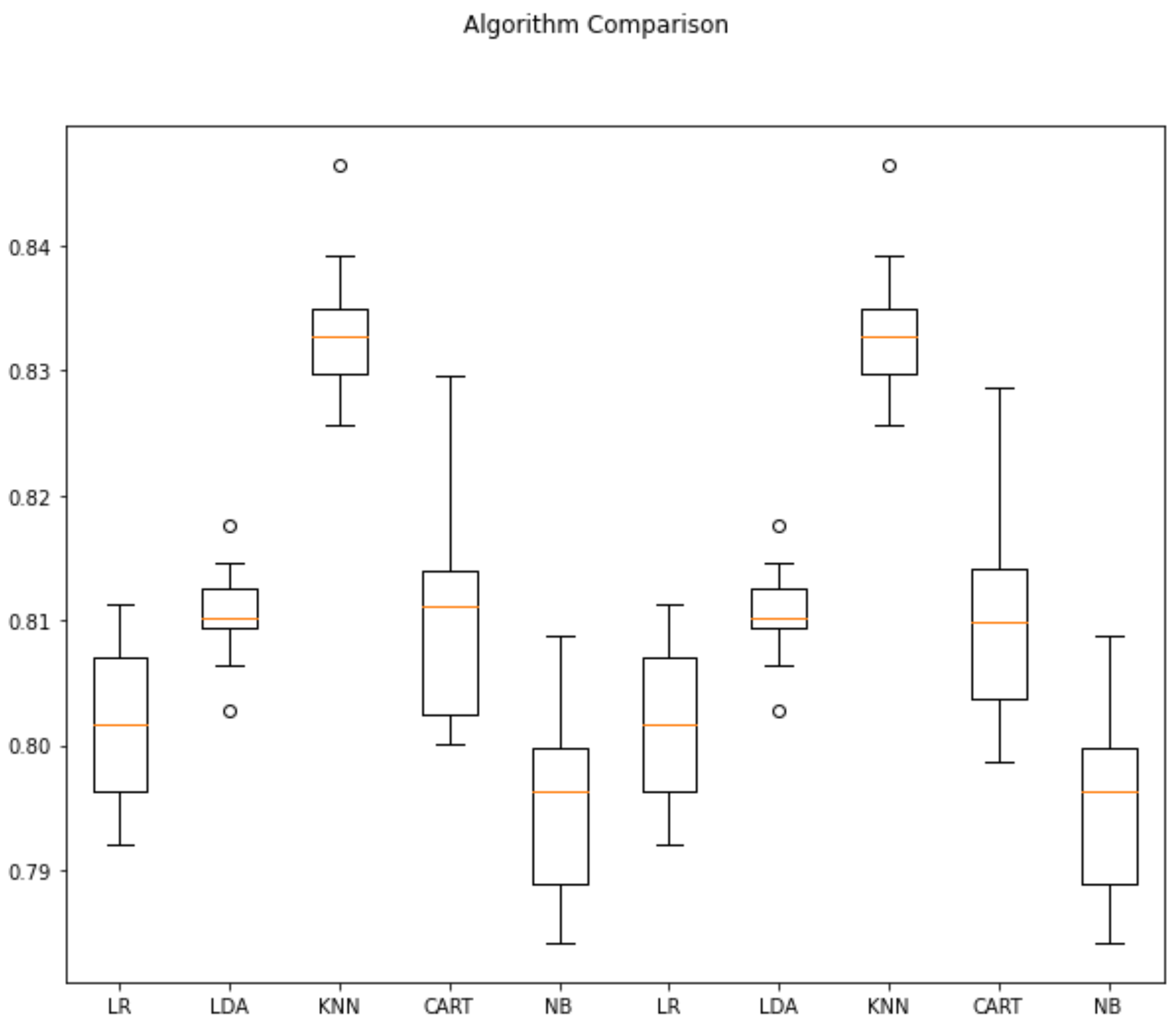
In [69]:

```
for name, model in models:
        kfold = model_selection.KFold(n_splits=10, random_state=seed,shuffl
        cv_results = model_selection.cross_val_score(model, x_train, y_trai
        results.append(cv_results)
        names.append(name)
        msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
        print(msg)
```

```
LR: 0.801532 (0.006570)
LDA: 0.810451 (0.003933)
KNN: 0.833228 (0.005868)
CART: 0.810119 (0.008247)
NB: 0.795498 (0.007394)
```

In [73]:

```
fig =plt.figure(figsize=(10,8))
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```



In comparision KNN has the best Accuracy

```
In [ ]:
```