Import Necessary Libraries

In [190…

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import model_selection
```

Business problem

Prepare a classification model using Naive Bayes for salary data

Data Collection

In [191…

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

In [192…

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

In [193…

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

Out[193…

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

In [194…

```
train.head()
```

Out[194…

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

In [195…

```
str_c = ["workclass","education","maritalstatus","occupation","relati
```

In [196…

```
number = LabelEncoder()
```

In [197…

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

In [199…

```
test.head()
```

Out[199…

	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 [200…

```
train.head()
```

Out[200…

	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 [201…

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

In [202…

```
train = train.replace({'Salary': mapping})
test = test.replace({'Salary': mapping})
```

In [203…

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

In [204…

```
salary=salary_data.copy()
```

In [205…

```
salary.head()
```

Out[205…

	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 [206…

```
salary.shape
```

Out[206… (45221, 14)

In [207…

```
salary.describe().T
```

Out[207…

	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 [208…

```
salary.isna().sum()
```

Out[208…

```
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
```

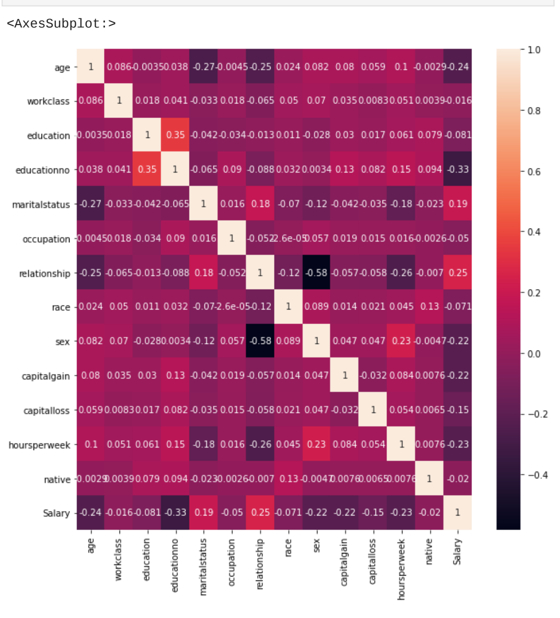
In [209…

```
corr = salary.corr()
```

In [210…

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

Out[210… <AxesSubplot:>



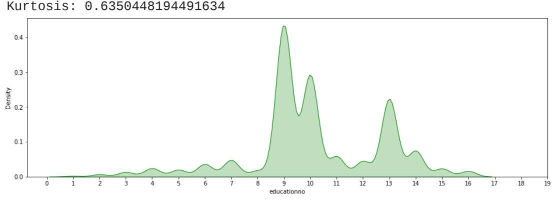
In [211…

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

In [212…

```
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
```



In [213…

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

In [214…

```
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>
```

In [165…

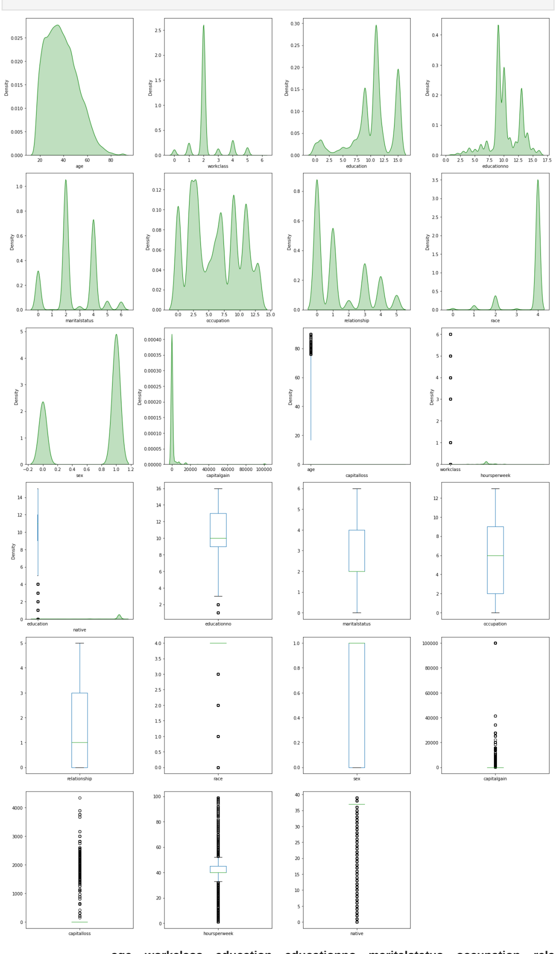
```
num_columns = dfa.select_dtypes(exclude='object').columns.tolist()
```

In [166…

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

In [113…

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



Out[113…

	age	workclass	education	educationno	maritalstatus	occupation	rela
skewness	0.532784	1.148931	-0.945666	-0.310621	-0.006760	0.107141	0
kurtosis	-0.155931	2.329983	0.773506	0.635045	-0.538981	-1.249883	-0

Naive Bayes

In [168…

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

GaussianNB

In [169…

```
clsfrgnb = GaussianNB()
```

In [170…

```
clsfrgnb.fit(x_train, y_train)
```

Out[170… GaussianNB()

In [171…

```
y_pred_gnb = clsfrgnb.predict(x_test)
```

In [172…

```
confusion_matrix(y_test, y_pred_gnb)
```

Out[172…

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

In [173…

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

Out[173…

col_0	GaussianNB()
row_0	
1	3700
2	11360

In [174…

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

```
Accuracy 0.7946879150066402
```

MultinomialNB

In [175…

```
clsfrmnb = MultinomialNB()
```

In [176…

```
clsfrmnb.fit(x_train, y_train)
```

Out[176… MultinomialNB()

In [177…

```
y_pred_mnb = clsfrmnb.predict(x_test)
```

In [178…

```
confusion_matrix(y_test, y_pred_mnb)
```

Out[178…

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

In [179…

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

Out[179…

col_0	MultinomialNB()
row_0	
1	3700
2	11360

In [180…

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

```
Accuracy 0.7749667994687915
```

Cross Validation

In [181…

```
seed = 7
```

In [182…

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

In [183…

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

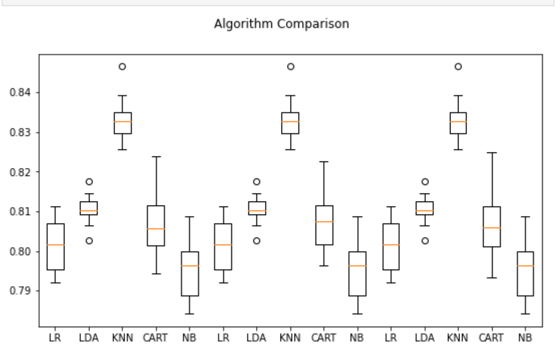
In [187…

```
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, shuffle=True, random_s
    cv_results = model_selection.cross_val_score(model, x_train, y_tr
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
LR: 0.801466 (0.006559)
LDA: 0.810451 (0.003933)
KNN: 0.833228 (0.005868)
CART: 0.806770 (0.008454)
NB: 0.795498 (0.007394)
```

In [215…

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



In [ ]:

In [ ]: