Lab 11:

1. Splitting into x and y

```
x = data.iloc[:, 0:-1]
x.shape
y = data.iloc[:,-1]
y.shape
```

2. Converting Object columns into Int columns

```
cat_columns = x.select_dtypes(['object']).columns
x[cat_columns] = x[cat_columns].apply(lambda x: pd.factorize(x)[0])
```

3. Splitting Data into train and test

```
X_train, X_test, Y_train, Y_test = train_test_split(x,y, test_size=.3, shuffle=False)
```

4. Applying Different Classifiers

a. Bernoulli

```
BernNB = BernoulliNB()
BernNB.fit(X_train,Y_train)
Y_bpred = BernNB.predict(X_test)

b_accuracy = metrics.accuracy_score(Y_test, Y_bpred)
print('Accuracy: %f' % b_accuracy)
b_precision = metrics.precision_score(Y_test, Y_bpred, average='weighted', labels=np.unique(Y_bpred))
print('Precision: %f' % b_precision)
b_recall = metrics.recall_score(Y_test, Y_bpred, average='weighted', labels=np.unique(Y_bpred))
print('Recall: %f' % b_recall)
b_f1 = metrics.f1_score(Y_test, Y_bpred, average='weighted', labels=np.unique(Y_bpred))
print('F1 score: %f' % b_f1)
```

b. Random Forest

```
RF = RandomForestClassifier()
RF.fit(X_train,Y_train)
Y_rpred = RF.predict(X_test)

r_accuracy = accuracy_score(Y_test, Y_rpred)
print('Accuracy: %f' % r_accuracy)
r_precision = metrics.precision_score(Y_test, Y_rpred, average='weighted', labels=np.unique(Y_rpred))
print('Precision: %f' % r_precision)
r_recall = metrics.recall_score(Y_test, Y_rpred, average='weighted', labels=np.unique(Y_rpred))
print('Recall: %f' % r_recall)
r_f1 = metrics.f1_score(Y_test, Y_rpred, average='weighted', labels=np.unique(Y_rpred))
print('F1 score: %f' % r_f1)
```

c. GaussianNB

```
GausNB = GaussianNB()
GausNB.fit(X_train,Y_train)
Y_gpred = GausNB.predict(X_test)

g_accuracy = accuracy_score(Y_test, Y_gpred)
print('Accuracy: %f' % g_accuracy)
g_precision = metrics.precision_score(Y_test, Y_gpred, average='weighted', labels=np.unique(Y_gpred))
print('Precision: %f' % g_precision)
g_recall = metrics.recall_score(Y_test, Y_gpred, average='weighted', labels=np.unique(Y_gpred))
print('Recall: %f' % g_recall)
g_f1 = metrics.f1_score(Y_test, Y_gpred, average='weighted', labels=np.unique(Y_gpred))
print('F1 score: %f' % g_f1)
```

d. Decision Tree

```
Dtree = DecisionTreeClassifier()
```

```
Dtree.fit(X_train,Y_train)
Y_dpred = Dtree.predict(X_test)

d_accuracy = accuracy_score(Y_test, Y_dpred)
print('Accuracy: %f' % d_accuracy)
d_precision = metrics.precision_score(Y_test, Y_dpred, average='weighted', labels=np.unique(Y_dpred))
print('Precision: %f' % d_precision)
d_recall = metrics.recall_score(Y_test, Y_dpred, average='weighted', labels=np.unique(Y_dpred))
print('Recall: %f' % d_recall)
d_f1 = metrics.f1_score(Y_test, Y_dpred, average='weighted', labels=np.unique(Y_dpred))
print('F1 score: %f' % d_f1)
```

e. MultinomialNB

```
MultiNB = MultinomialNB()
MultiNB.fit(X_train,Y_train)
Y_mpred = MultiNB.predict(X_test)

m_accuracy = accuracy_score(Y_test, Y_mpred)
print('Accuracy: %f' % m_accuracy)
m_precision = metrics.precision_score(Y_test, Y_mpred, average='weighted', labels=np.unique(Y_mpred))
print('Precision: %f' % m_precision)
m_recall = metrics.recall_score(Y_test, Y_mpred, average='weighted', labels=np.unique(Y_mpred))
print('Recall: %f' % m_recall)
m_f1 = metrics.f1_score(Y_test, Y_mpred, average='weighted', labels=np.unique(Y_mpred))
print('F1 score: %f' % m_f1)
```

f. KNN

```
KNN = KNeighborsClassifier()
KNN.fit(X_train,Y_train)
Y_kpred = KNN.predict(X_test)

k_accuracy = accuracy_score(Y_test, Y_kpred)
print('Accuracy: %f' % k_accuracy)
k_precision = metrics.precision_score(Y_test, Y_kpred, average='weighted', labels=np.unique(Y_kpred))
print('Precision: %f' % k_precision)
k_recall = metrics.recall_score(Y_test, Y_kpred, average='weighted', labels=np.unique(Y_kpred))
print('Recall: %f' % k_recall)
k_f1 = metrics.f1_score(Y_test, Y_kpred, average='weighted', labels=np.unique(Y_kpred))
print('F1 score: %f' % k_f1)
```

5. Plotting Line Graph Showcasing all the Scores of applied Classifiers

```
plt.figure(figsize=(16,10))
#plot 1:
x1 = np.array(['Bernoulli', 'Random Forest', 'Gaussian', 'Decision Tree', 'Multinomial', 'KNeighbors'])
y1 = np.array([b_accuracy, r_accuracy, g_accuracy, d_accuracy, m_accuracy, k_accuracy])
plt.plot(x1,y1, marker = 'o', label='Accuracy')

#plot 2:
x1 = np.array(['Bernoulli', 'Random Forest', 'Gaussian', 'Decision Tree', 'Multinomial', 'KNeighbors'])
y1 = np.array([b_precision, r_precision, g_precision, d_precision, m_precision, k_precision])
plt.plot(x1,y1, marker = 'o', label='Precision')

#plot 3:
x1 = np.array(['Bernoulli', 'Random Forest', 'Gaussian', 'Decision Tree', 'Multinomial', 'KNeighbors'])
y1 = np.array([b_recall, r_recall, g_recall, d_recall, m_recall, k_recall])
plt.plot(x1,y1, marker = 'o', label='Recall')

#plot 4:
x1 = np.array(['Bernoulli', 'Random Forest', 'Gaussian', 'Decision Tree', 'Multinomial', 'KNeighbors'])
y1 = np.array([b_f1, r_f1, g_f1, d_f1, m_f1, k_f1])
plt.plot(x1,y1, marker = 'o', label='F1')

plt.title("Scores of Applied Classifiers")
plt.legend()
plt.show()
```

6. Plotting Bar Graph showcasing all the scores of applied classifiers

```
# x-coordinates of left sides of bars
left = [1, 2, 3, 4, 5, 6]

plt.figure(figsize=(16,10))
# heights of bars
height = [b_f1, r_f1, g_f1, d_f1, m_f1, k_f1]

# labels for bars
tick_label = ['Bernoulli', 'Random Forest', 'Gaussian', 'Decision Tree', 'Multinomial', 'KNeighbors']

# plotting a bar chart
plt.bar(left, height, tick_label = tick_label, width = 0.9, color = ['#08737f', '#00898a', '#089f8f', '#39b48e', '#64c987',
    '#92dc7e'])

# naming the x-axis
plt.xlabel('Classifiers')
# naming the y-axis
plt.ylabel('F1 Scores')
# plot title
plt.title('F1 Scores of Applied Classifiers')

# function to show the plot
plt.show()
```

Lab 11 Task

1. **(Continue from Lab 10):**
   - Train test Splitting
   - Choose and apply the right model
   - Testing/Predicting
   - Display Accuracy Score