Classification Using Naive Bayes and Tree Based Algorithm

Concept Session

Demo 4.1 : Classification Using Naive Bayes Algorithm

Import libraries

In [ ]:
```
import pandas as pd
import numpy as np
from numpy.random import default_rng
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
from matplotlib.colors import ListedColormap
from sklearn.metrics import precision_score, recall_score, accuracy_score
```

Data Collection

In [ ]:
```
rng = default_rng()
vals = rng.standard_normal(10)

# for each input column: get the number of counts, the user plays
features = ['number of clouds', 'temperature', 'humidity', 'wind speed']
n_samples = 100

n_clouds = 3 + rng.standard_normal(n_samples)*1
temperatures_numeric = 20 + rng.standard_normal(n_samples) * 2
humidity = 50 + rng.standard_normal(n_samples) * 20
wind_speed = 15 + rng.standard_normal(n_samples) * 5

X = np.array([n_clouds, temperatures_numeric, humidity, wind_speed]).T
print(X[:10])

# do not play, if the number of clouds is larger than 3, if the humidity is higher than 80 % and if the
 wind speed is larger than 20 km/h
y = np.ones((X.shape[0]))
y[X[:, 0] > 3] = 0
y[X[:, 2] > 80] = 0
y[X[:, 3] > 20] = 0
print(y[:10])
df = pd.DataFrame(list(zip(n_clouds, temperatures_numeric, humidity, wind_speed, y)), columns = ['numbe
r of clouds', 'temperature', 'humidity', 'wind speed', 'play'])
df.head()
```

Data Encoding

In [ ]:
```
def get_encoded_df(df, columns=None):
    if columns == None:
        columns = df.columns

    for col in columns:
        le = LabelEncoder()
        df[col] = le.fit_transform(df[col])

    return df


df_encoded = get_encoded_df(df)
df_encoded.head()
```

Data Splitting

In [ ]:
```
x = df_encoded.iloc[:, [1,2]].values
y = df_encoded.iloc[:, 4].values
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 0)
#print(x_train)
#print(x_test)
```

Data Scaling

In [ ]:
```
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

Build the model

In [ ]:
```
# Fitting Naive Bayes to the Training set
classifier = GaussianNB()
classifier.fit(x_train, y_train)
```

Evaluate the Model

In [ ]:
```
# Predicting the Test set results
y_pred = classifier.predict(x_test)
```

In [ ]:
```
# Making the Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

In [ ]:
```
accuracy_score(y_test, y_pred)
```

Data Visualization

In [ ]:
```
x_set, y_set = x_train, y_train
X1, X2 = np.meshgrid(np.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step = 0.01
), np.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape), alpha
 = 0.75, cmap = ListedColormap(('lightblue', 'lightgreen')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1], c = ListedColormap(('blue', 'green'))(i), l
abel = j)
plt.title('Naive Bayes (Training set)')
plt.xlabel('Temperature')
plt.ylabel('Humidity')
plt.legend()
plt.show()
```

In [ ]:
```
# Visualising the Test set results
x_set, y_set = x_test, y_test
X1, X2 = np.meshgrid(np.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step = 0.01
),
                     np.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01

))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('lightblue', 'lightgreen')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
                c = ListedColormap(('blue', 'green'))(i), label = j)
plt.title('Naive Bayes (test set)')
plt.xlabel('Temperature')
plt.ylabel('Humidity')
plt.legend()
plt.show()
```

Demo 4.2 : Classification Using Decision Tree Based Algorithm

Import Data & Python Packages

In [ ]:
```
# Load libraries
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
```

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

In [ ]:
```
# load dataset
pima = pd.read_csv("/content/drive/MyDrive/ML/DS2_C5_S4_Diabetes_Data_Concept.csv")
pima.columns = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age', 'label']
pima.head()
```

Data Exploration

In [ ]:
```
pima.dtypes
```

In [ ]:
```
#explore the numeric data types
pima.describe()
```

In [ ]:
```
#average numbers for all columns
pima.groupby('label').mean()
```

Preparation of Data

In [ ]:
```
#split dataset in features and target variable
feature_cols = ['pregnant', 'insulin', 'bmi', 'age','glucose','bp','pedigree']
X = pima[feature_cols] # Features
y = pima.label # Target variable
```

In [ ]:
```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1) # 70% training
 and 30% test
```

Modelling - Decision Tree Classifier

In [ ]:
```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(max_depth=3)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

Evaluation

In [ ]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

In [ ]:
```
# confusion matrix
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred)
```

In [ ]:
```
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score

print('accuracy:', accuracy_score(y_test, y_pred))
print('recall:', recall_score(y_test, y_pred, average='weighted'))
print('f1-score:', f1_score(y_test, y_pred, average='weighted'))
print('precision:', precision_score(y_test, y_pred))
```

Visualization

In [ ]:
```
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

# Setting dpi = 300 to make image clearer than default
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=300)

tree.plot_tree(clf,
               feature_names = feature_cols,
               filled = True);

fig.savefig('Diabetes_Tree.png')
```