```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import numpy as np
import pandas as pd
import math
from sklearn.model_selection import train_test_split
import sklearn.neighbors
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neighbors import KNeighborsRegressor
from sklearn import metrics
from sklearn.preprocessing import scale
from collections import Counter
```

K Nearest Neighbour Classifier

```
# Import dataset
df=pd.read_csv('/content/drive/MyDrive/MATERI/Pembelajaran Mesin/Praktikum Genap 20212022/iris.csv')
df.head()
```

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

```
# Memisahkan data menjadi 70:30 (train:test) pengujian
df_X=df.iloc[:,:4]
df_Y=df.iloc[:,4]
X_train,X_test,Y_train,Y_test=train_test_split(df_X,df_Y,test_size=0.3,random_state=33)
```

```
# Mengubah indeks catatan menjadi berurutan
X_train.index=range(len(X_train))
Y_train.index=range(len(X_train))
X_test.index=range(len(X_test))
Y_test.index=range(len(Y_test))
```

```
# Berfungsi untuk mengembalikan daftar jarak test record dari train record
def distNeighbours(X_train,Y_train,X_test,K):
    distance=[]
    for i in range(len(X_train)):
        eDistance=0
        for j in range(len(X_train.columns)):
                eDistance+=round(np.sqrt(pow((X_train.iloc[i,j]-X_test[j]),2)),2)
        distance.append((eDistance,i,Y_train.iloc[i]))
        distance=sorted(distance, key=lambda x: x[0])[0:K]
    return distance

# Memprediksi output dari variabel kategori berdasarkan K tetangga terdekat
# Output adalah kelas yang paling sering di antara K tetangga terdekat
def predictOutputCategorical(X_train,Y_train,X_test,K):
    neighbours=[]
    responses=[]
    for i in range(len(X_test)):
        neighbours.append(distNeighbours(X_train,Y_train,X_test.iloc[i,:],K))
    for i in neighbours:
        votes={}
        for j in i:
            if j[-1] in votes.keys():
                votes[j[-1]]=votes[j[-1]]+1
```

```
            else:
                votes[j[-1]]=1
        responses.append(sorted(votes,key=votes.get,reverse=True)[0])
    return responses

# Memprediksi output dari variabel numerik berdasarkan K tetangga terdekat
# Output adalah mean dari K tetangga terdekat
def predictOutputNumeric(X_train,Y_train,X_test,K):
    neighbours=[]
    responses=[]
    for i in range(len(X_test)):
        neighbours.append(distNeighbours(X_train,Y_train,X_test.iloc[i,:],K))
    for i in neighbours:
        mean=0
        for j in i:
            mean+=j[-1]
        mean=mean/K
        responses.append(mean)
    return responses

# Akurasi prediksi kategoris
def getAccuracyCategorical(actual,predicted):
    correct=0
    for i in range(len(predicted)):
        if predicted[i]==actual[i]:
            correct+=1
    return round((correct/len(actual))*100,2)

# Akurasi prediksi numerik
def getAccuracyNumeric(actual,predicted):
    error=0
    for i in range(len(predicted)):
        error+=pow((actual[i]-predicted[i]),2)
    error=error/len(predicted)-1
    return 100-error
```

```
# Predict species
output=predictOutputCategorical(X_train,Y_train,X_test,3)
getAccuracyCategorical(Y_test,output)
```

```
97.78
```

```
# Fit model using in built sklearn function
model=KNeighborsClassifier(n_neighbors=3,p=2,metric='minkowski')
model.fit(X_train,Y_train)
```

```
KNeighborsClassifier(n_neighbors=3)
```

```
# Accuracy of the model
print('Accuracy: {:^0.2f}'.format(metrics.accuracy_score(Y_test,model.predict(X_test))*100))
```

```
Accuracy: 97.78
```

```
# Check whether the both outputs are same or not
# They are same as displayed below
output==model.predict(X_test)
```

```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True])
```

K Nearest Neighbour Regression

```
# Import dataset
# This is for trying out regression using KNN
df=pd.read_csv('/content/drive/MyDrive/MATERI/Pembelajaran Mesin/Praktikum Genap 20212022/Freshmen.csv')
df.head()
```

	GPA	Miles from Home	College	Accommodations	Years Off	Part-Time Work Hours	Attends Office Hours	High School GPA
0	0.73	253	Social Sciences	Dorm	4	35	Sometimes	3.23
1	1.60	143	Social Sciences	Dorm	5	30	Never	2.35
2	2.17	171	Social Sciences	Dorm	0	25	Never	3.95

```
# Change the data types of the categorical variables accordingly
df.College=df.College.astype('category')
df.Accommodations=df.Accommodations.astype('category')
df['Attends Office Hours']=df['Attends Office Hours'].astype('category')
```

```
# Generate dummy values of the categorical variables and drop one (i.e. n-1 dummies for n categories)
df_dummies=pd.get_dummies(df,drop_first=True)
# Display top 5 records
df_dummies.head()
```

	GPA	Miles from Home	Years Off	Part-Time Work Hours	High School GPA	College_Engineering	College_Liberal Arts	College_Sciences	College_Social Sciences	Accomm
0	0.73	253	4	35	3.23	0	0	0	1	
1	1.60	143	5	30	2.35	0	0	0	1	
2	2.17	171	0	25	3.95	0	0	0	1	
3	1.02	332	5	30	3.44	0	0	1	0	
4	3.14	112	0	25	3.20	0	0	0	0	

```
# Specifying the X and Y
X_train=df_dummies.iloc[:,1:]
Y_train=df_dummies.GPA
```

```
# Splitting data into 70:30 train:test ratio
X_train,X_test,Y_train,Y_test=train_test_split(X_train,Y_train,test_size=0.3,random_state=33)
```

```
# Changing the index of the records to sequential
X_train.index=range(len(X_train))
Y_train.index=range(len(X_train))
X_test.index=range(len(X_test))
Y_test.index=range(len(Y_test))
```

```
# Predict GPA
output=predictOutputNumeric(X_train,Y_train,X_test,3)
print('Accuracy from the code: {:^0.2f}'.format(getAccuracyNumeric(Y_test,output),2))
```

```
Accuracy from the code: 99.94
```

```
model=KNeighborsRegressor(n_neighbors=3,p=2)
model.fit(X_train,Y_train)
```

```
KNeighborsRegressor(n_neighbors=3)
```

```
print('Accuracy from the model {:^0.2f}'.
      format(metrics.mean_squared_error(Y_test,model.predict(X_test))*100))
```

```
Accuracy from the model 99.88
```

```
# Check whether both the outputs are same or not
# They are not same - Need to find why?
output==model.predict(X_test)
```

```
array([ True,  True,  True,  True,  True,  True, False,  True, False,
        True,  True,  True,  True,  True, False,  True,  True,  True,
       False,  True,  True, False,  True, False,  True,  True,  True,
        True, False, False])
```

✓ 0s completed at 3:03 PM