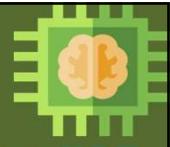


Elective Course

Course Code: CS4103

Autumn 2025-26

**Lecture #35**

Artificial Intelligence for Data Science

Week-10:**MACHINE LEARNING (Part III)****Exploring K-Nearest Neighbor (KNN) using Python****Course Instructor:****Dr. Monidipa Das**

Assistant Professor

Department of Computational and Data Sciences

Indian Institute of Science Education and Research Kolkata, India 741246

Iris Dataset from Scikit-learn Library



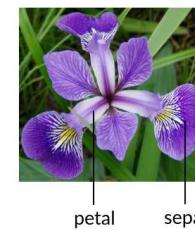
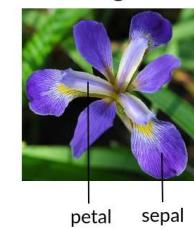
```
from sklearn.datasets import load_iris

# Load the Iris dataset
iris = load_iris()

# Accessing the data and target
X = iris.data
y = iris.target

# Accessing feature and target names
feature_names = iris.feature_names
target_names = iris.target_names

print("Input features (X) shape:", X.shape)
print("Target (y) shape:", y.shape)
print("Input feature names:", feature_names)
print("Target names:", target_names)
```

**iris setosa****iris versicolor****iris virginica**

```
Input features (X) shape: (150, 4)
Target (y) shape: (150,)
Input feature names: ['sepal length (cm)', 'sepal width
(cm)', 'petal length (cm)', 'petal width (cm)']
Target names: ['setosa' 'versicolor' 'virginica']
```

KNN on Iris: Dataset Loading



```

import numpy as np
import pandas as pd

# Loading the dataset
dataset= pd.read_csv('F:/CS4103/iris.csv')

#view dataset
print("First 6 rows of the dataset:")
print(dataset.head(6))

#Dataset column names
print("\nName of the columns in the dataset:")
print(list(dataset.columns))
input_features = list(dataset.columns[:-1])
print("Name of the input feature columns:",input_features)
class_col=dataset.columns[-1]
print("Name of the class/label column:",class_col)

```

First 6 rows of the dataset:

	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
5	5.4	3.9	1.7	0.4	Setosa

Name of the columns in the dataset:
['sepal.length', 'sepal.width', 'petal.length', 'petal.width',
'variety']

Name of the input feature columns: ['sepal.length',
'sepal.width', 'petal.length', 'petal.width']

Name of the class/label column: variety

Dr. Monidipa Das, Department of CDS, IISER Kolkata

KNN on Iris: Exploring Dataset Details



```

# Number of rows or instances and number of columns features
print("\nTotal number of data
points/examples/instances:",
dataset.shape[0])
print("Total number of input features:"
dataset.shape[1]-1)

# Dataset description
print("\nDescription of the dataset:")
print(dataset.describe())

# Print number of classes
c=dataset[class_col].nunique()
print("\nNumber of classes (discrete
labels):", c)

# Number of instances per class
print("\nSample count per class:")
print(dataset[class_col].value_counts())

```

Total number of data points/examples/instances: 150
Total number of input features: 4

Description of the dataset:

	sepal.length	sepal.width	petal.length	petal.width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Number of classes (discrete labels): 3

Sample count per class:

Class	Count
Setosa	50
Versicolor	50
Virginica	50

Name: variety, dtype: int64

Dr. Monidipa Das, Department of CDS, IISER Kolkata

KNN on Iris: Data Preprocessing and Data Split



```

x = dataset[input_features].values
y = dataset[class_col].values

#KNeighborsClassifier does not accept string/categorial labels. We need to
#transform them into numbers
print("\nClasses before encoding:")
print(np.unique(y))

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)

print("\nClasses after encoding:")
print(np.unique(y))

#split dataset into training set and test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 0)

```

Classes before encoding:
['Setosa' 'Versicolor' 'Virginica']
Classes after encoding:
[0 1 2]

Dr. Monidipa Das, Department of CDS, IISER Kolkata

KNN on Iris: Data Visualization



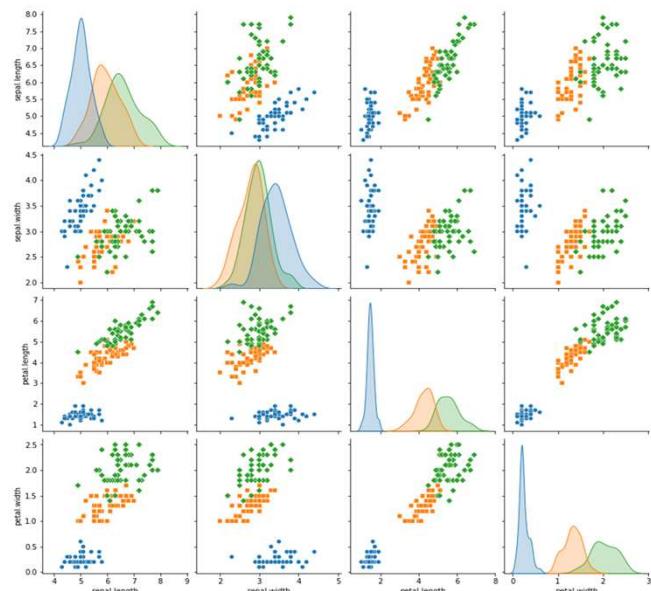
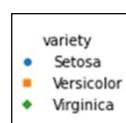
```

#visualize the distribution of features
#and their relationship with others
import matplotlib.pyplot as plt

import seaborn as sns
plt.figure()
sns.pairplot(dataset, hue = class_col,
size=3, markers=["o", "s", "D"])

plt.show()

```



Dr. Monidipa Das, Department of CDS, IISER Kolkata

KNN on Iris: Building KNN Classifier



```
#Using KNN for classification
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score,
ConfusionMatrixDisplay

# Instantiate learning model (K = 3)
classifier = KNeighborsClassifier(n_neighbors=3)

# Fitting the model
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

#Evaluating predictions
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]
```

Dr. Monidipa Das, Department of CDS, IISER Kolkata

KNN on Iris: Evaluating KNN Classifier

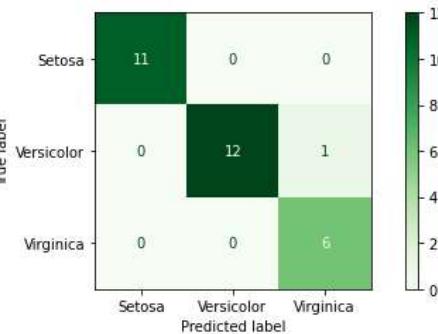


```
#Evaluating predictions (contd.)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=dataset[class_col].unique())
disp.plot(cmap='Greens')
plt.grid(False)
plt.show()

accuracy = accuracy_score(y_test, y_pred)

print('Accuracy of our model is equal ' +
str(round(accuracy, 2)) + '%.')

from sklearn.metrics import classification_report
# A comprehensive classification report
report = classification_report(y_test, y_pred)
print("\nClassification Report:\n", report)
```



Classification Report:				
	precision	recall	f1-score	support
0	1.00	1.00	1.00	11
1	1.00	0.92	0.96	13
2	0.86	1.00	0.92	6
accuracy				0.97
macro avg	0.95	0.97	0.96	30
weighted avg	0.97	0.97	0.97	30

Dr. Monidipa Das, Department of CDS, IISER Kolkata

KNN on Iris: Finding Best K using Cross Validation Score



```

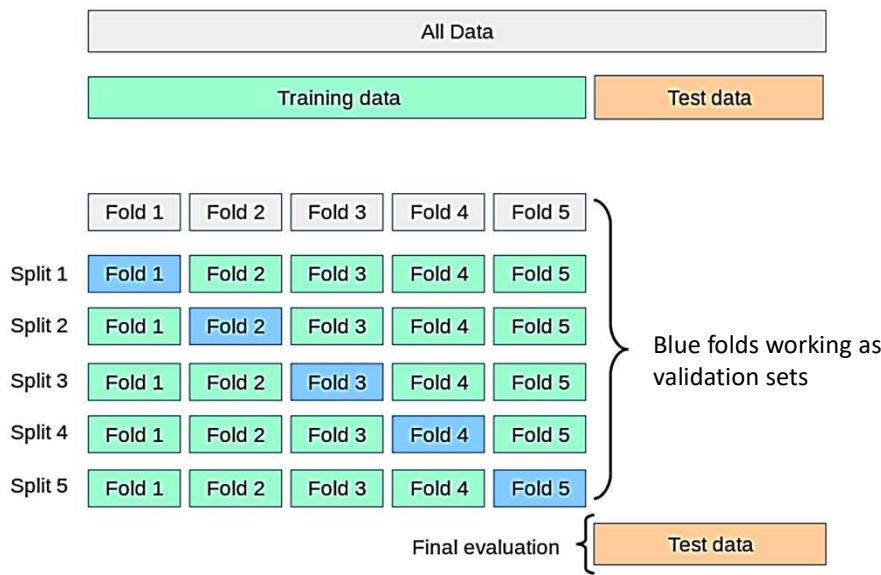
from sklearn.model_selection import cross_val_score
k_list = list(range(1,30,2)) # creating list of K for KNN
cv_scores = [] # creating list of cv scores
# perform 10-fold cross validation
for k in k_list:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X_train, y_train, cv=10, scoring='accuracy')
    cv_scores.append(scores.mean())

plt.figure(figsize=(15,10))
plt.title('Finding optimal value of K', fontsize=30, fontweight='bold')
plt.xlabel('Number of Neighbors K', fontsize=25)
plt.ylabel('Accuracy', fontsize=25)
plt.plot(k_list, cv_scores, linewidth=5, color='g')
plt.xticks(k_list, fontsize=20)
plt.yticks(fontsize=20)
plt.show()

```

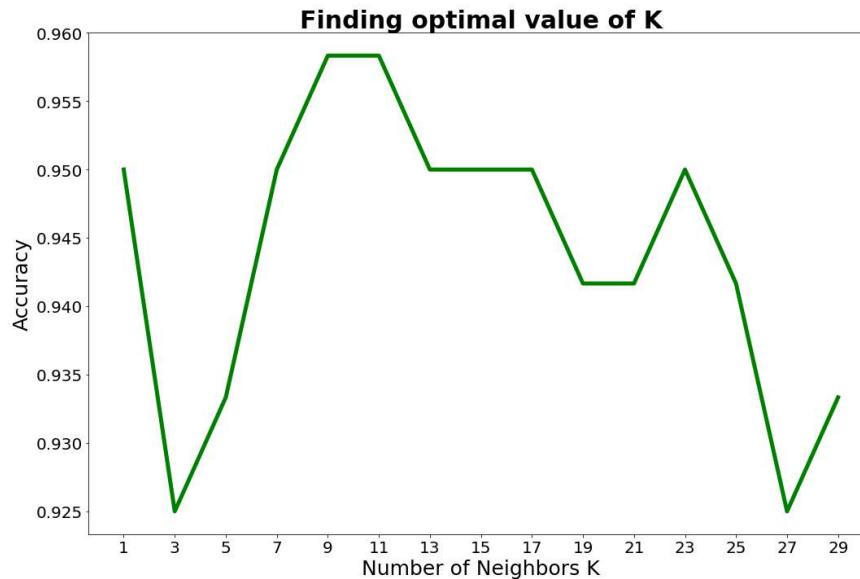
Dr. Monidipa Das, Department of CDS, IISER Kolkata

Cross Validation



Dr. Monidipa Das, Department of CDS, IISER Kolkata

KNN on Iris: Finding Best K using Cross Validation Score



Dr. Monidipa Das, Department of CDS, IISER Kolkata

KNN on Iris: Classifier using Best K

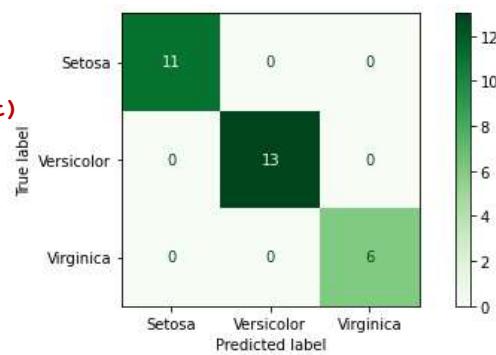


```

best=k_list[cv_scores.index(max(cv_scores))]
print("Best value of K:",best)
# Instantiate learning model with best K
classifier = KNeighborsClassifier(n_neighbors=best)
# Fitting the model
classifier.fit(X_train, y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test)

#Evaluating predictions
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=dataset[class_col].unique())
disp.plot(cmap='Greens')
plt.grid(False)
plt.show()
accuracy = accuracy_score(y_test, y_pred)*100
print('Accuracy of our model is equal ' + str(round(accuracy, 2)) + ' %.')

```



```

Best value of K: 9
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
Accuracy of our model is equal 100.0 %.

```

Dr. Monidipa Das, Department of CDS, IISER Kolkata

KNN on Iris: Visualizing KNN Decision Boundary



```

from matplotlib.colors import ListedColormap

cmap_light = ListedColormap(['lightskyblue','lightcoral', 'lightgreen'])
cmap_bold = ['blue','red', 'green']

X_train=X_train[:,[0,1]]
clf = KNeighborsClassifier(best)
clf.fit(X_train, y_train)

h=0.01
x_min, x_max = X_train[:, 0].min() - 1, X_train[:, 0].max() + 1
y_min, y_max = X_train[:, 1].min() - 1, X_train[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

c=[]
for label in y:
    c.append(cmap_bold[label])

```

Nearest neighbor algorithm does not explicitly compute decision boundaries, but these can be inferred

Dr. Monidipa Das, Department of CDS, IISER Kolkata

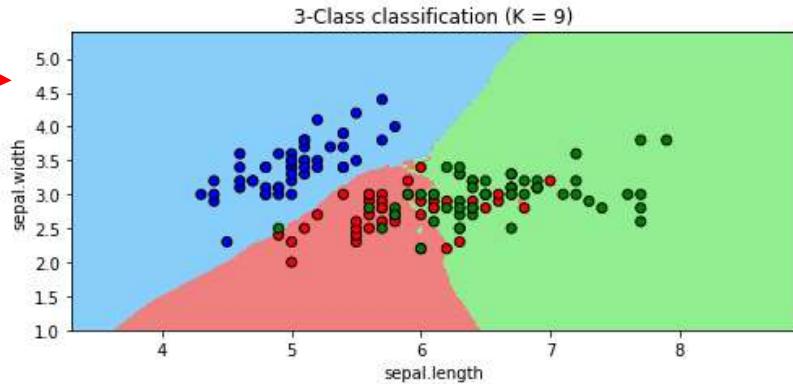
KNN on Iris: Visualizing KNN Decision Boundary



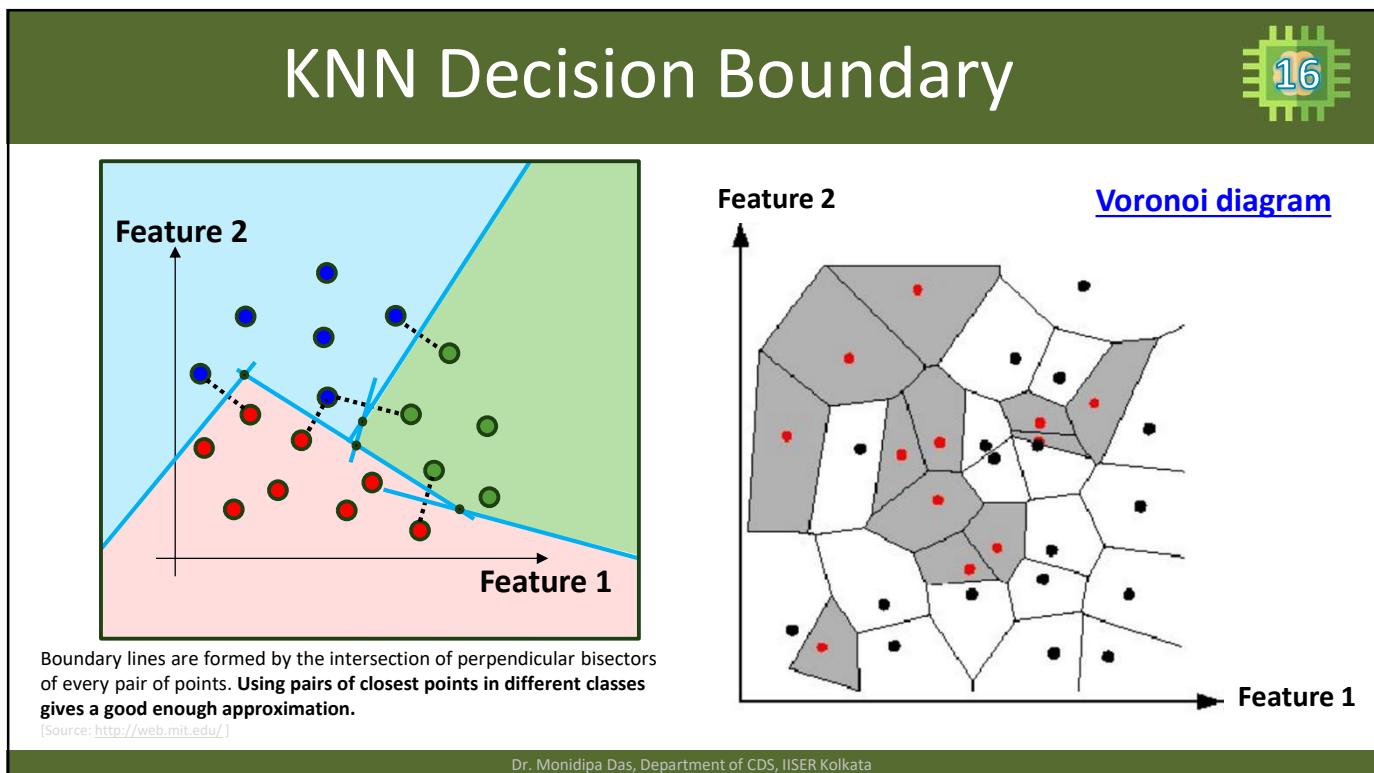
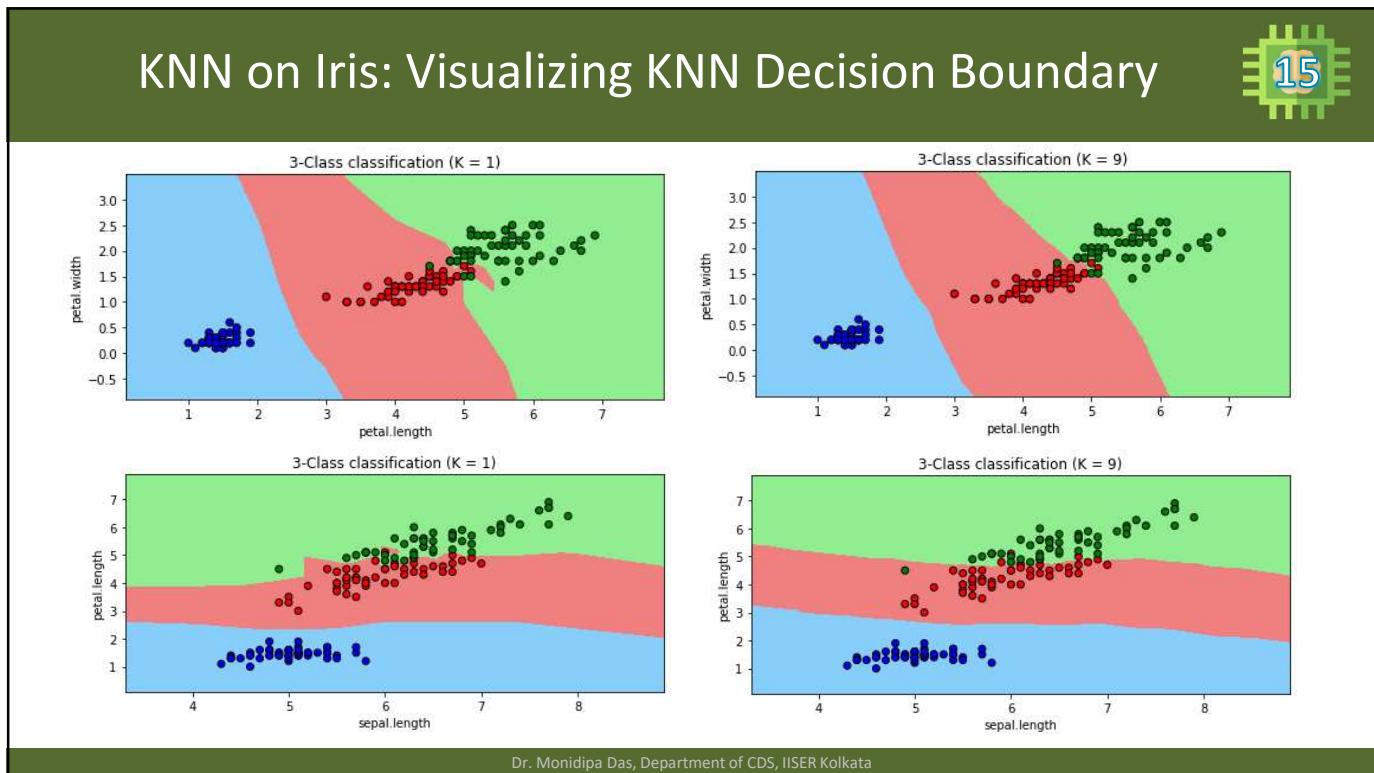
```

plt.figure()
plt.contourf(xx, yy, Z, cmap=cmap_light)
plt.scatter(x=X[:, 0], y=X[:, 1], color=c, edgecolors="black")
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.title("3-Class classification (K = %i)" % (best))
plt.xlabel(input_features[0])
plt.ylabel(input_features[1])
plt.show()

```



Dr. Monidipa Das, Department of CDS, IISER Kolkata



KNN and Impact of Feature Scaling



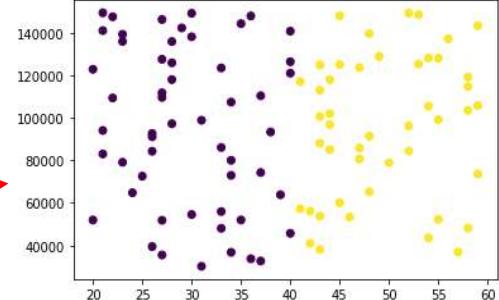
```

import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

# Create a synthetic dataset
np.random.seed(42)
ages = np.random.randint(20, 60, 100).reshape(-1, 1)
salaries = np.random.randint(30000, 150000, 100)
    .reshape(-1, 1)
target = np.where(ages > 40, 1, 0)

X = np.hstack((ages, salaries))
y = target
plt.figure()
plt.scatter(X[:,0], X[:,1], c=y[:,0])

```



Dr. Monidipa Das, Department of CDS, IISER Kolkata

KNN and Impact of Feature Scaling



```

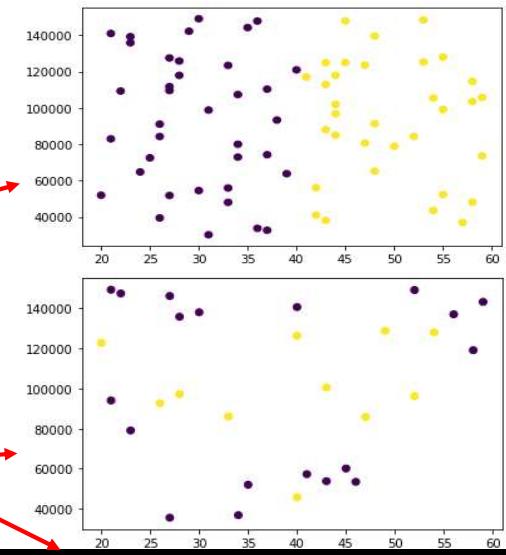
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X,
y, test_size=0.3, random_state=42)

# Scenario 1: Without Standardization
knn_no_scale = KNeighborsClassifier(n_neighbors=5)
knn_no_scale.fit(X_train, y_train)
plt.figure()
plt.scatter(X_train[:,0], X_train[:,1], c=y_train[:,0])

y_pred_no_scale = knn_no_scale.predict(X_test)
accuracy_no_scale = accuracy_score(y_test,
y_pred_no_scale)

print(f"Accuracy without standardization:
{accuracy_no_scale:.2f}")
plt.figure()
plt.scatter(X_test[:,0], X_test[:,1], c=y_pred_no_scale)

```



Dr. Monidipa Das, Department of CDS, IISER Kolkata

KNN and Impact of Feature Scaling



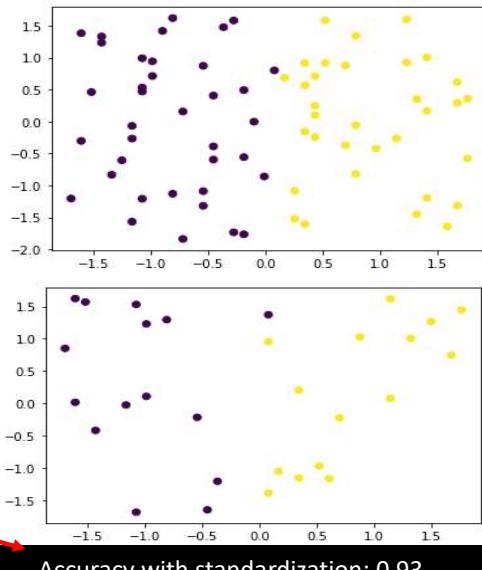
```
#Scenario 2: With Standardization
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

knn_scaled = KNeighborsClassifier(n_neighbors=5)
knn_scaled.fit(X_train_scaled, y_train)

plt.figure()
plt.scatter(X_train_scaled[:,0], X_train_scaled[:,1],
c=y_train[:,0])

y_pred_scaled = knn_scaled.predict(X_test_scaled)
accuracy_scaled = accuracy_score(y_test,
y_pred_scaled)

print(f"Accuracy with standardization:
{accuracy_scaled:.2f}")
plt.figure()
plt.scatter(X_test_scaled[:,0], X_test_scaled[:,1],
c=y_pred_scaled)
```



Dr. Monidipa Das, Department of CDS, IISER Kolkata

KNN: Pros and Cons



- **Pros:**
 - Simple to understand and easy to implement.
 - With zero training time, it can be a useful tool
 - KNN works just as easily with multiclass data sets
 - Non-parametric nature of KNN

- **Cons:**
 - Computationally expensive testing phase
 - Must store all training data
 - KNN can suffer from skewed class distributions.
 - Accuracy of KNN can be severely degraded with high-dimension data

Dr. Monidipa Das, Department of CDS, IISER Kolkata

KNN: Summary



- Instance-based Learning
- Non-parametric
- Naturally forms complex decision boundaries
- KNN works well with lots of samples
- Sensitive to class noise
- Sensitive to scales of attributes
- Distances are less meaningful in high dimensions
- Expensive at run time

Dr. Monidipa Das, Department of CDS, IISER Kolkata



Questions?

Dr. Monidipa Das, Department of CDS, IISER Kolkata