Project Title	iris classification
Tools	Jupyter Notebook and VS code
Technologies	Machine learning
Domain	Data Analytics
Project Difficulties level	Advanced

Dataset : Dataset is available in the given link. You can download it at your convenience.

Click here to download data set

Project Overview

The Iris Classification project involves creating a machine learning model to classify iris flowers into three species (Setosa, Versicolour, and Virginica) based on the length and width of their petals and sepals. This is a classic problem in machine learning and is often used as an introductory example for classification algorithms.

Problem Statement:

- The model should achieve a high level of accuracy in classifying iris species.
- The model's predictions should be consistent and reliable, as measured by cross-validation.
- The final report should provide clear and comprehensive documentation of the project, including all code, visualizations, and findings.

By achieving these objectives, the project will demonstrate the ability to apply machine learning techniques to a classic classification problem, providing insights into the characteristics of different iris species and the effectiveness of various algorithms for this task.

Project Steps

Understanding the Problem

- The goal is to classify iris flowers into one of three species based on four features: sepal length, sepal width, petal length, and petal width.

Dataset Preparation

- **Dataset**: The Iris dataset is available in the UCI Machine Learning Repository and is also included in many machine learning libraries, such as scikit-learn.
- **Features**: Sepal length, sepal width, petal length, petal width.
- **Labels**: Iris species (Setosa, Versicolour, Virginica).

Data Exploration and Visualization

- Load the dataset and explore it using descriptive statistics and visualization techniques.
- Use libraries like Pandas for data manipulation and Matplotlib/Seaborn for visualization.
- Example visualizations include scatter plots, pair plots, and histograms to understand the distribution and relationships between features.

Data Preprocessing

- Handle missing values (if any).
- Standardize or normalize the features if necessary to ensure they are on a similar scale.
- Split the dataset into training and testing sets (commonly 80% training and 20% testing).

Model Selection and Training

- Choose a classification algorithm. Common choices for this problem include:
    - K-Nearest Neighbors (KNN)
    - Decision Trees
    - Random Forest
    - Support Vector Machine (SVM)
    - Logistic Regression
- Train the model using the training data.

Model Evaluation

- Evaluate the model using the testing data.
- Use metrics like accuracy, precision, recall, and F1-score to assess the model's performance.
- Visualize the confusion matrix to understand the classification results in detail.

Hyperparameter Tuning

- Use techniques like Grid Search or Random Search to find the optimal hyperparameters for the chosen model.
- Cross-validation can also be employed to ensure the model generalizes well to unseen data.

Model Interpretation and Insights

- Interpret the model results and understand which features are most important for the classification.
- Visualize decision boundaries if using models like Decision Trees or SVM.

Deployment (Optional)

- Deploy the model using a web framework like Flask or Django to create a simple web application where users can input flower measurements and get predictions.

Documentation and Reporting

- Document the entire process, including data exploration, preprocessing, model training, evaluation, and tuning.
- Create a final report or presentation summarizing the project, results, and insights.

Content

It includes three iris species with 50 samples each as well as some properties about each flower. One flower species is linearly separable from the other two, but the other two are not linearly separable from each other.

The columns in this dataset are:

Id
SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm
Species
Sepal Width vs. Sepal Length

Sample Code

Here's a basic example using Python and scikit-learn to classify iris species:

```
# Import necessary libraries
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
iris = load_iris()
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['species'] = iris.target

# Explore the dataset
print(df.head())
print(df.describe())
sns.pairplot(df, hue='species')
plt.show()
```

```
# Split the data into training and testing sets
X = df.drop('species', axis=1)
y = df['species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Train the model
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)

# Make predictions
y_pred = knn.predict(X_test)

# Evaluate the model
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

# Plot confusion matrix
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

This code demonstrates loading the Iris dataset, splitting it into training and testing sets, standardizing the features, training a KNN classifier, and evaluating its performance.

Additional Tips

- Experiment with different algorithms to see which one performs best for your dataset.
- Use dimensionality reduction techniques like PCA (Principal Component Analysis) to visualize high-dimensional data.
- Perform feature engineering if you believe additional features could improve model performance.

Sample Project Report

In [2]:

```
df = pd.read_csv('../input/iris-classification/Iris.csv')
df.head()
```

Out[2]:

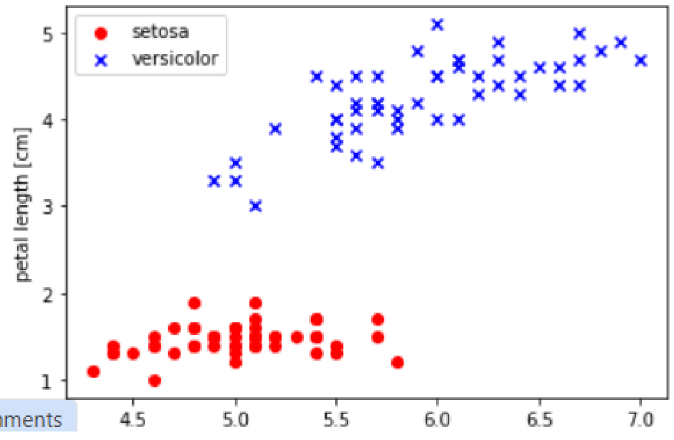
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [3]:

```
y = df.iloc[0:100, 4].values
y = np.where(y == 'Iris-setosa', -1, 1)
X = df.iloc[0:100, [0, 2]].values
```

In [4]:

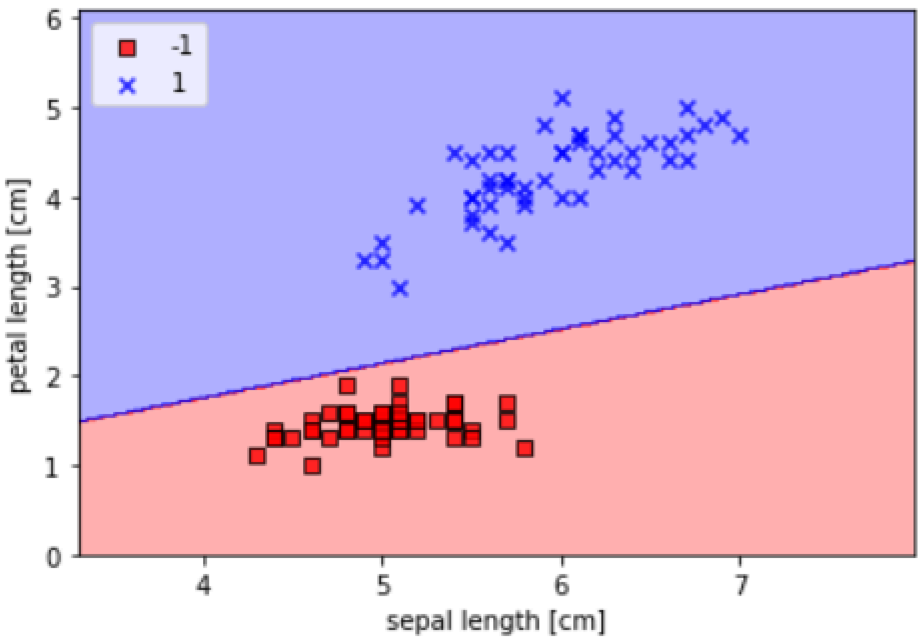
```
plt.scatter(X[:50, 0], X[:50, 1], color='red', marker='o', label='setosa')
plt.scatter(X[50:100,0], X[50:100,1], color='blue', marker='x', label='versicolor')
plt.xlabel('sepal length [cm]')
plt.ylabel('petal length [cm]')
plt.legend(loc='upper left')
plt.show()
```



ification/comments

[10]:

```
plot_decision_regions(X, y, classifier=ppn)
plt.xlabel('sepal length [cm]')
plt.ylabel('petal length [cm]')
plt.legend(loc='upper left')
plt.show();
```


petal length [cm]
sepal length [cm]