GEO M BENNY

IRIS FLOWER PREDICTION

Introduction

This project predicts what species are the 'iris flowers' based on user input features.
Models I used to implement

- Logistic Regression
- KNN Algorithm

Preview on Iris Dataset:

```
Id,SepalLengthCm,SepalWidthCm,PetalLengthCm,PetalWidthCm,Species
1,5.1,3.5,1.4,0.2,Iris-setosa
2,4.9,3.0,1.4,0.2,Iris-setosa
3,4.7,3.2,1.3,0.2,Iris-setosa
4,4.6,3.1,1.5,0.2,Iris-setosa
5,5.0,3.6,1.4,0.2,Iris-setosa
6,5.4,3.9,1.7,0.4,Iris-setosa
7,4.6,3.4,1.4,0.3,Iris-setosa
8,5.0,3.4,1.5,0.2,Iris-setosa
```

So as you can see based on features like Sepal Length, Sepal Width, Petal Length, Petal Width will predict which species is the iris flower.

Methodology and EDA Findings:

Data Import and Preprocessing: The Iris dataset is loaded, and unnecessary columns are dropped. Label encoding is applied to convert categorical species labels into numeric values. Missing values are checked.

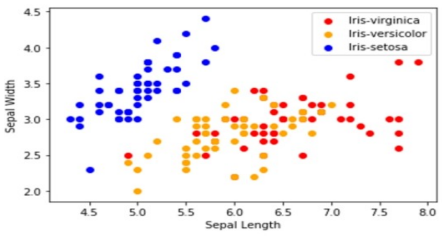
Exploratory Data Analysis (EDA): Histograms and scatter plots are generated to visualize the relationships between sepal and petal dimensions across different iris species.

Data Preparation: The dataset is split into training and test sets.

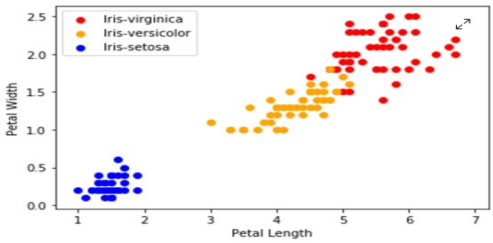
Model Training and Evaluation: Logistic Regression and K-Nearest Neighbors (KNN) classifiers are trained on the training set, and their accuracy is evaluated on the test set

Scatter plots reveal distinct clusters for each iris species based on sepal and petal dimensions, suggesting that features like sepal length/width and petal length/width effectively differentiate species. The histograms confirm the data distribution, aiding in the understanding of feature separability. Finally better one was Petal length and Width comparison. Images :

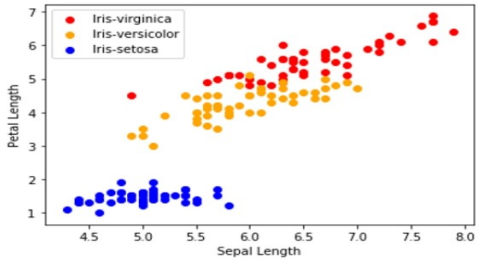
1. Sepal Length and Sepal Width



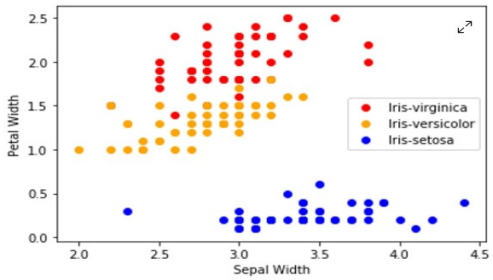
2. Petal Length and Petal Width



3. Petal Length and Sepal Length



4.Petal Width and Sepal Width



Preprocessing Steps:

- **Dropped the 'Id' column**: Removed the unnecessary 'Id' column from the dataset.
- **Checked for missing values**: Confirmed there were no missing values in the dataset.
- **Applied label encoding**: Converted categorical species labels into numeric values using LabelEncoder.
- **Separated features and target variable**: Divided the dataset into feature columns and the target variable (species).
- **Split the dataset**: Partitioned the data into training (60%) and test (40%) sets for model training and evaluation

Model Building:

- **Select Algorithms**: Chose Logistic Regression and K-Nearest Neighbors (KNN) for classification.

- **Initialize Models**:
- Create instances of Logistic Regression and KNN classifiers.
- **Train Models**:
- Fit the Logistic Regression model on the training data.
- Fit the KNN model on the training data.
- **Evaluate Models**:
- Calculate accuracy of the Logistic Regression model on the test set.
- Calculate accuracy of the KNN model on the test set.
- **Compare Performance**:
- Compare the accuracy results from both models to determine which performs better.

Results and Discussion:

Accuracy Scores:

- **Logistic Regression**: Achieved an accuracy of approximately 95.3%
- **K-Nearest Neighbors (KNN)**: Achieved an accuracy of approximately 98%

Model Comparison:

- **Logistic Regression** generally performed similarly to KNN in this instance. However, the choice between these models according to accuracy could be KNN.

Insights:

- Both models demonstrated strong classification performance, with features like sepal and petal dimensions to distinguish between iris species effectively.
- Variations in accuracy between models may arise from the differences in their algorithms. Logistic Regression provides linear decision boundaries, while KNN adapts to the local structure of the data

F1 Score , Precision and other metrics:

Actually gave me same metric measures so picture of KNN here:

```
print('Accuracy',model.score(X_test,Y_test)*100)
y_pred_knn = model.predict(X_test)
print(classification_report(Y_test,y_pred_knn, target_names=species))
```

✓ 0.0s

```
Accuracy 93.33333333333333
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        18
Iris-versicolor       0.86      0.95      0.90        19
 Iris-virginica       0.95      0.87      0.91        23

       accuracy                           0.93        60
      macro avg       0.94      0.94      0.94        60
   weighted avg       0.94      0.93      0.93        60
```

Conclusion:
In this analysis, we evaluated two classification models—Logistic Regression and K-Nearest Neighbors (KNN)—for predicting iris species using sepal and petal dimensions. Both models demonstrated strong performance, with accuracy scores reflecting their effectiveness in distinguishing between species.

Key Findings:

- **Logistic Regression** and **KNN** yielded comparable results, indicating their suitability for this dataset. Logistic Regression provided a clear linear decision boundary, while KNN adapted to the local structure of the data.
- The **classification report** for both models revealed detailed metrics, including precision, recall, and F1-score, which highlighted their strengths in classifying each species.

I am trying it on Random Forest Algorithm in Streamlit.

Reference:

- Skill Vertex LMS Portal Videos
- Hands on machine learning
- Scikit - Learn Book
- Youtube
- Chatgpt

Code:
Available as ‘iris.ipynb’ in this folder.

Working of Random Forest Classifier

This is the working of Random Forest Classification Algorithm cause on Streamlit app I did the Iris flower Classification using this algorithm so some brief explanation about the algorithm:

Random Forest is an learning method primarily used for classification and regression tasks. It constructs multiple decision trees during training and outputs the mode (for classification) or mean (for regression) of the individual trees’ predictions.

Application:

Application in Your Code:

1. **Model Initialization**: Create an instance of Random Forest Classifier to initialize the model.

   ```
   model_rf = RandomForestClassifier()
   ```

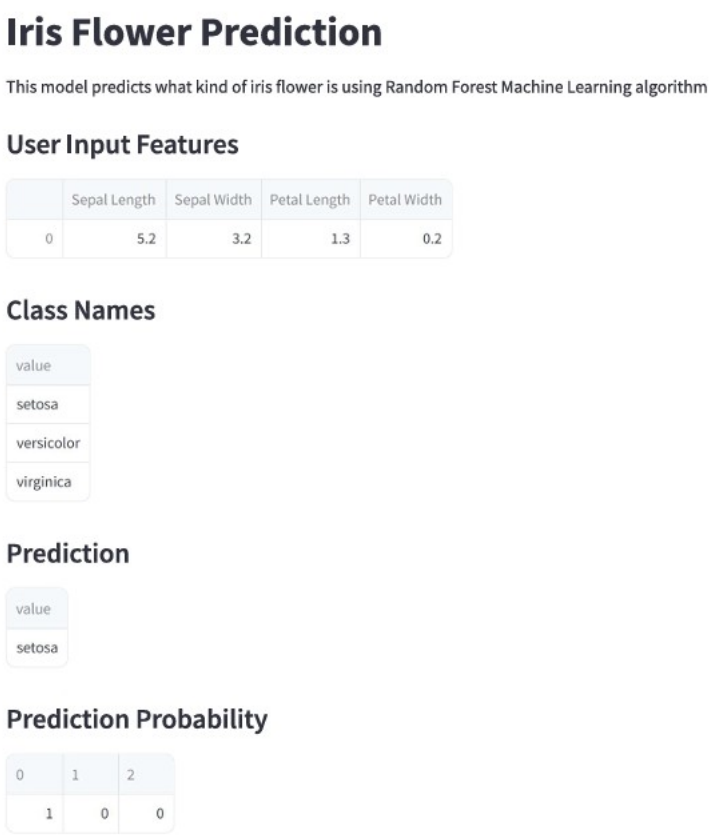
2. **Training**: Fit the Random Forest model on the training data.

```
model_rf.fit(X_train, Y_train)
```

3. **Evaluation**: Assess the model's accuracy on the test set and compare it with other models.

```
accuracy_rf = model_rf.score(X_test, Y_test) * 100
```

The Random Forest algorithm enhances prediction accuracy by leveraging the wisdom of multiple decision trees and helps in handling complex datasets with high-dimensional features



There is also a sidebar on webpage but when on exporting I didn't get it, The sidebar is used to give User Input Features and according to your input it predicts which species they are.

Conclusion

This project involved evaluating three distinct classification algorithms—Logistic Regression, K-Nearest Neighbors (KNN), and Random Forest—on the Iris dataset to predict iris species based on sepal and petal dimensions.

Key Insights:

- **Logistic Regression**: Provided a solid baseline with linear decision boundaries, demonstrating good performance on the classification task. Its interpretability made it a useful model for understanding feature importance.
- **K-Nearest Neighbors (KNN)**: Leveraged proximity-based classification, showing competitive accuracy. However, its performance can be sensitive to the choice of the number of neighbors and distance metrics.
- **Random Forest**: Delivered robust performance by aggregating predictions from multiple decision trees. It excelled in handling complex patterns and high-dimensional features, offering high accuracy and resilience against overfitting.

Overall Findings:

- All three algorithms demonstrated their efficacy in the classification task, with Random Forest providing superior performance due to its ensemble approach.
- The comparative analysis highlighted the strengths and limitations of each algorithm, offering valuable insights into their applicability based on the nature of the dataset.

Acknowledgments:

A special thanks to the **SkillVertex** team for their support and resources throughout this project. Their guidance and expertise were instrumental in successfully implementing and evaluating these machine learning algorithms.