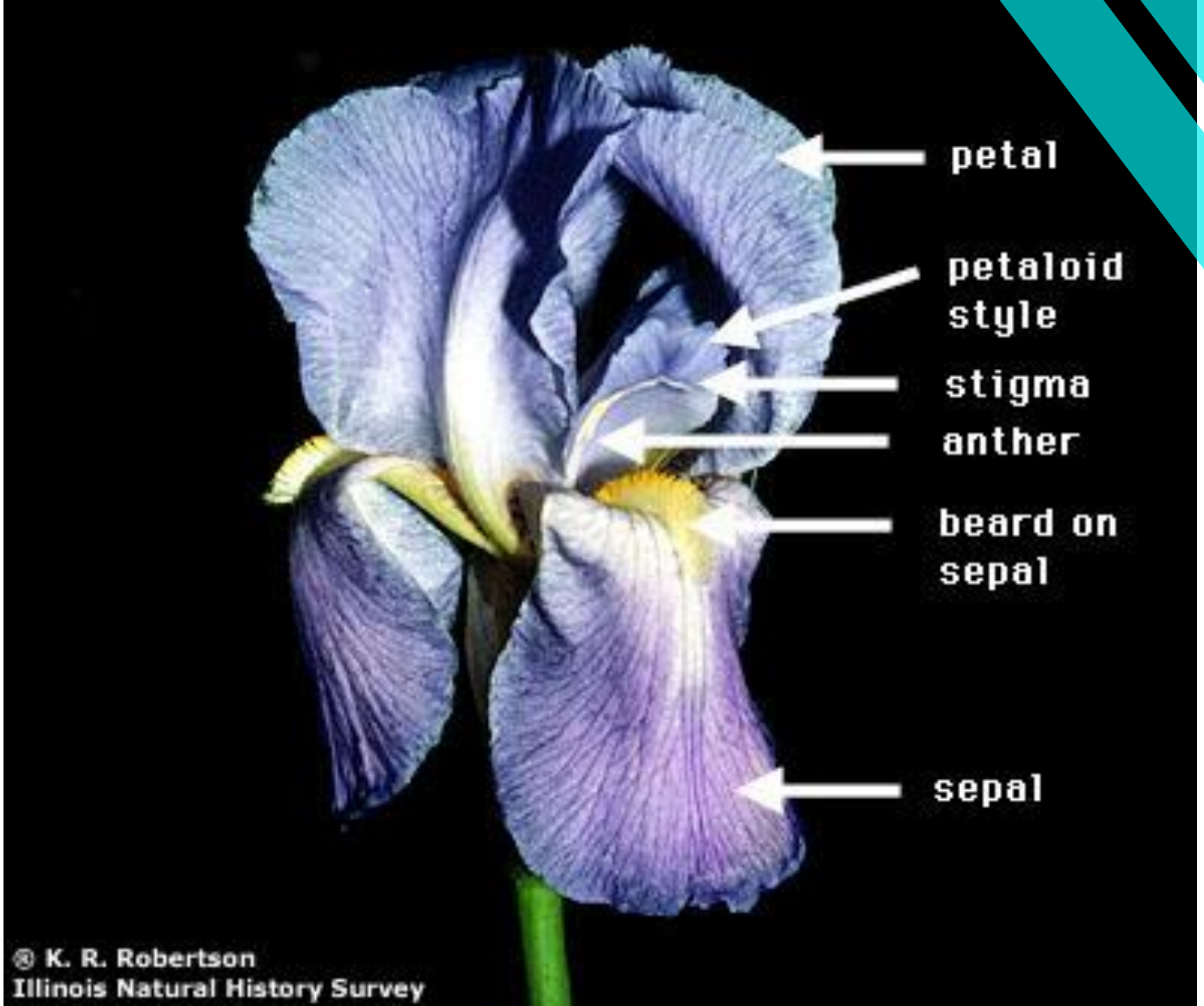
Iris Flower Dataset

Ultra-junior: Vera Georgiana-Florentina

https://www.kaggle.com/datasets?feedbackIds=8

About Dataset

The Iris flower data set, introduced by Ronald Fisher in 1936, is a well-known multivariate data set used in statistical and machine learning analysis. Collected by Edgar Anderson, it includes measurements of four features (sepal length, sepal width, petal length, and petal width) from 150 samples of three species of Iris flowers: Iris setosa, Iris virginica, and Iris versicolor.


petal
petaloid
style
stigma
anther
beard on
sepal
sepal
© K. R. Robertson
Illinois Natural History Survey

Analiza exploratorie a datelor (EDA)

Încărcarea Setului de Date

Cu ajutorul comenzii pd.read_csv() am încărcat setul de date Iris Flower

DATASETS

iris-flower-dataset

Inspecția Datelor

Forma, nr. și conținutul rândurilor și coloanelor, tipul de date și nr. pentru fiecare specie sunt inspectate

	sepal_length	sepal_width	petal_length	petal_width	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

Gestionarea Valorilor Lipsă și a Duplicatelor

Mai întâi, am verificat valorile lipsă și rândurile duplicate. După aceea, am eliminat intrările duplicate.

```
iris.isnull().sum()
```

```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

```
iris.duplicated().sum()
```

```
3
```

+ Code + Markdown

There are no null values in the dataset.

There are 3 duplicate entries. I am dropping them.

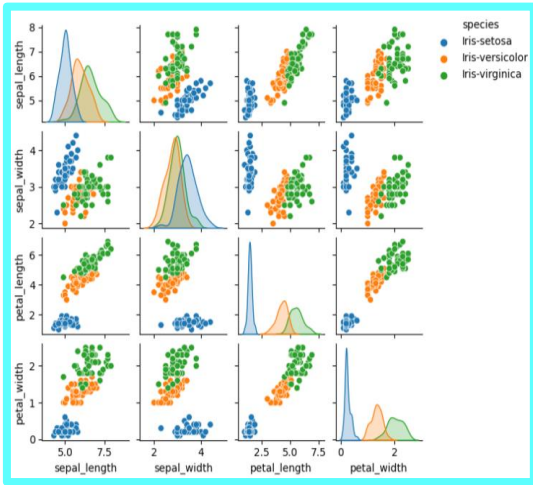
Statistică Descriptivă

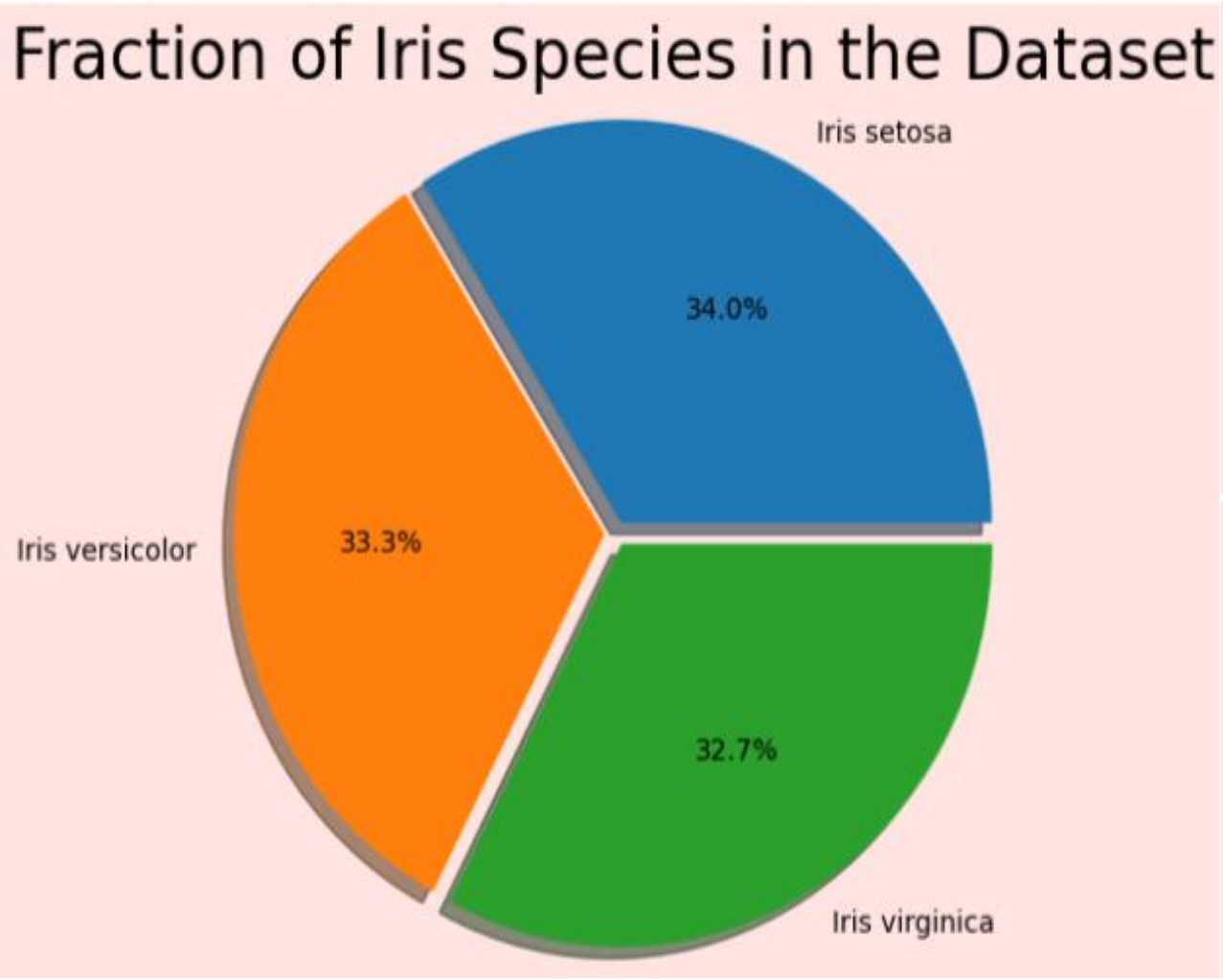
Statisticile sunt implementate pentru a rezuma și organiza datele astfel încât acestea să poată fi ușor înțelese.

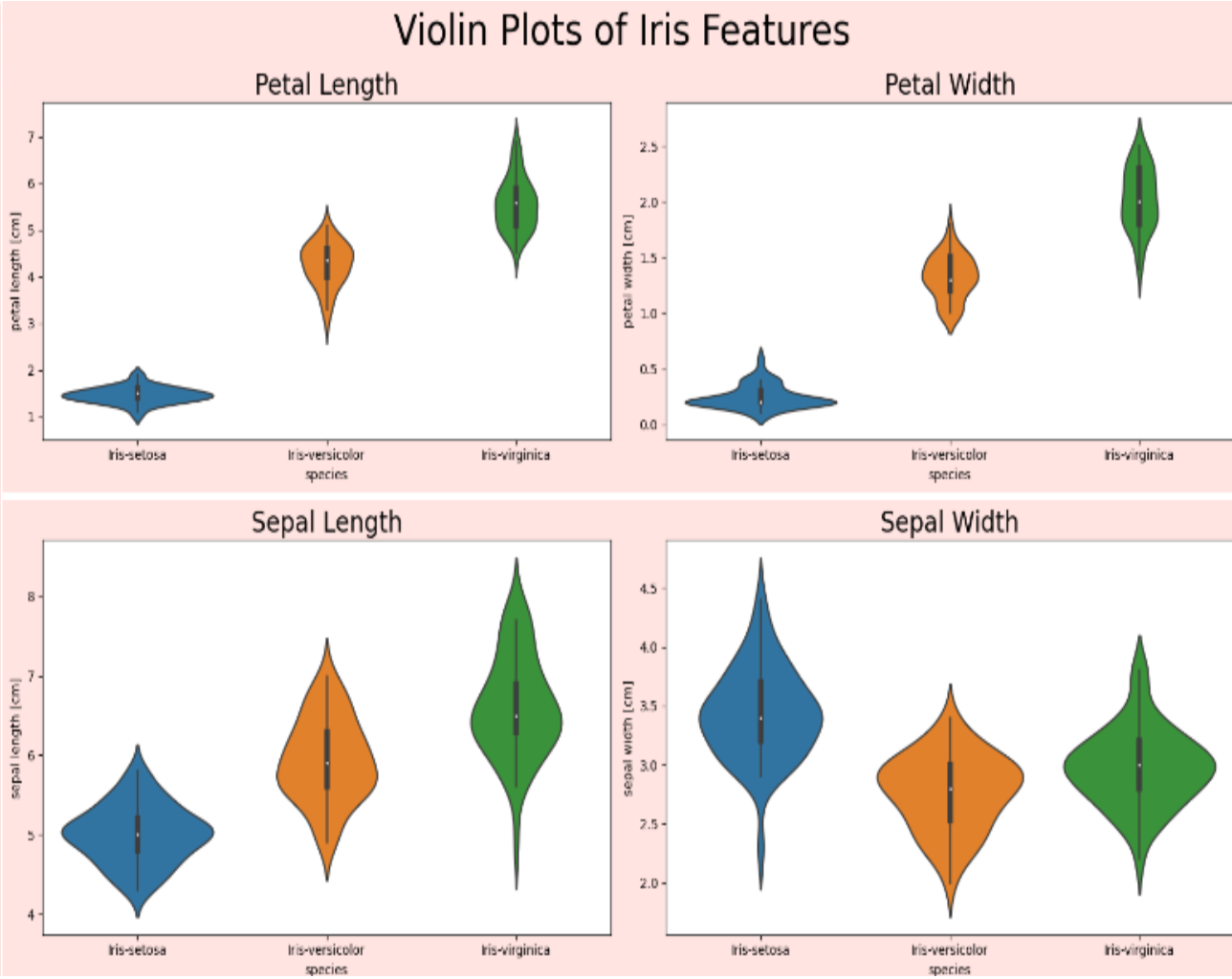
	sepal_length	sepal_width	petal_length	petal_width
count	147.0	147.0	147.0	147.0
mean	5.856	3.056	3.78	1.209
std	0.8291	0.437	1.759	0.7579
min	4.3	2.0	1.0	0.1
25%	5.1	2.8	1.6	0.3
50%	5.8	3.0	4.4	1.3
75%	6.4	3.3	5.1	1.8
max	7.9	4.4	6.9	2.5

Vizualizări

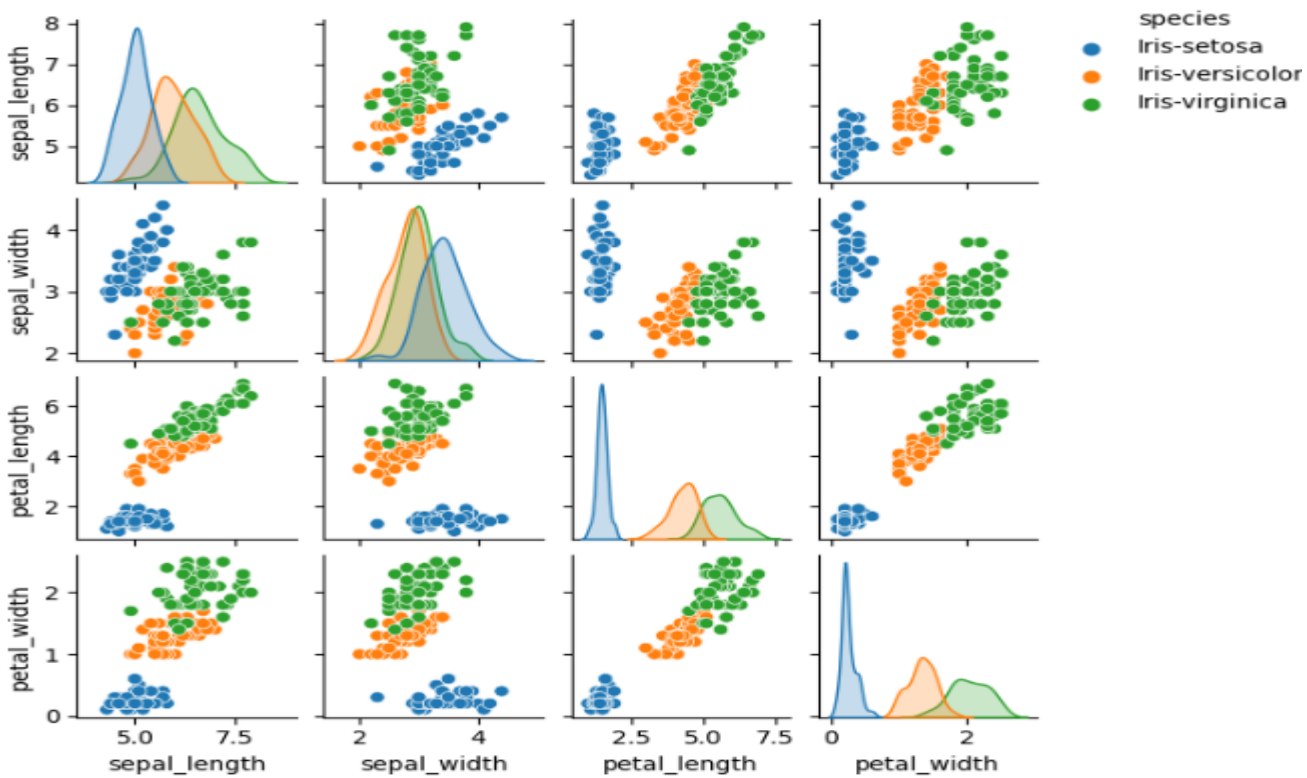
Sunt vizualizate mai multe grafice pentru a vedea distribuția datelor și relația dintre caracteristici

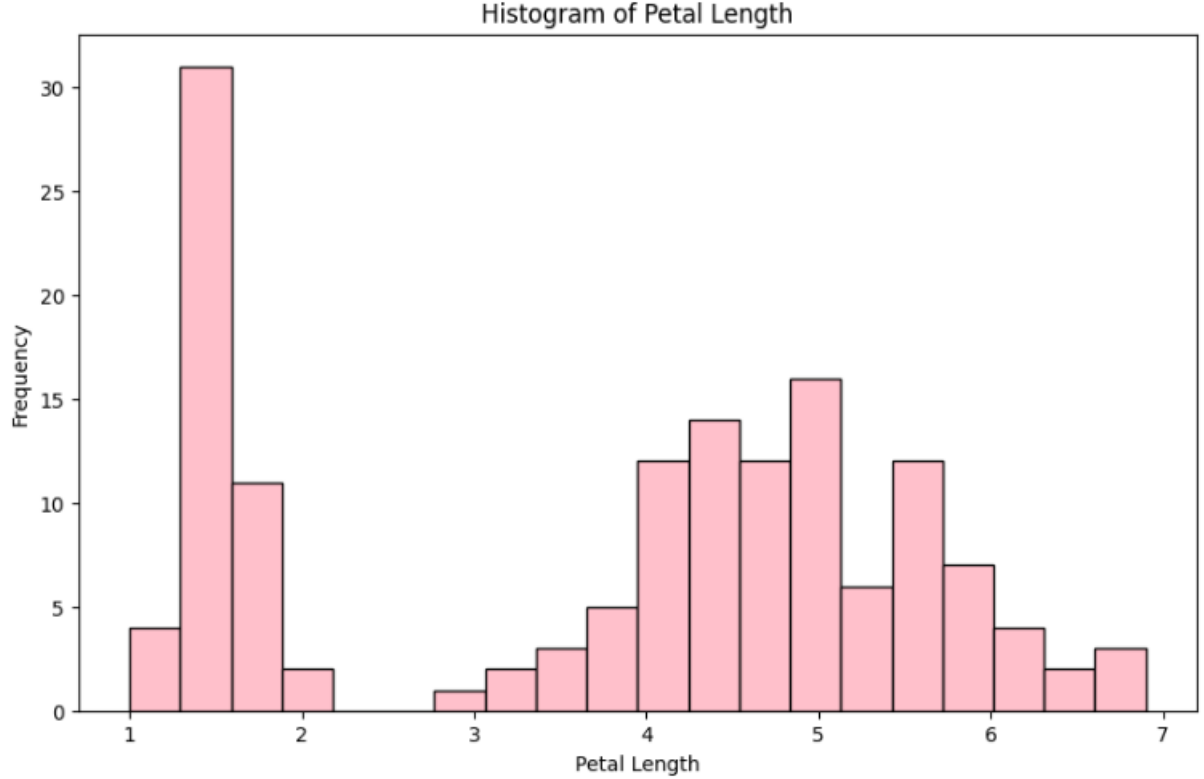


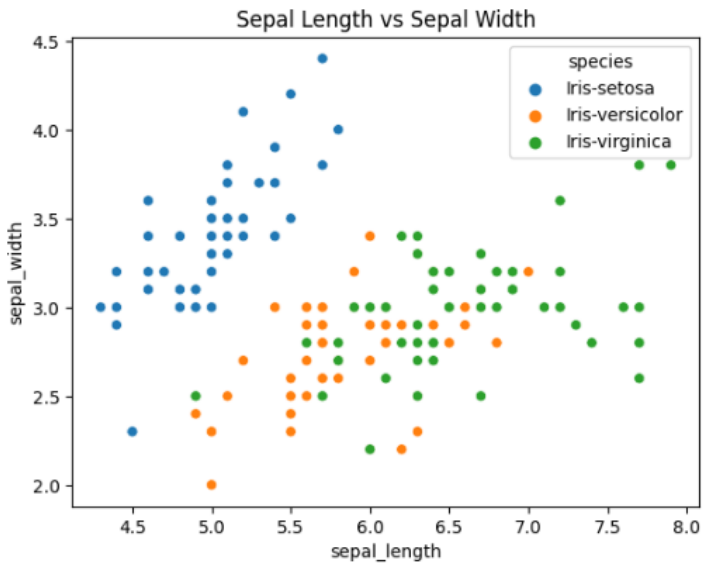


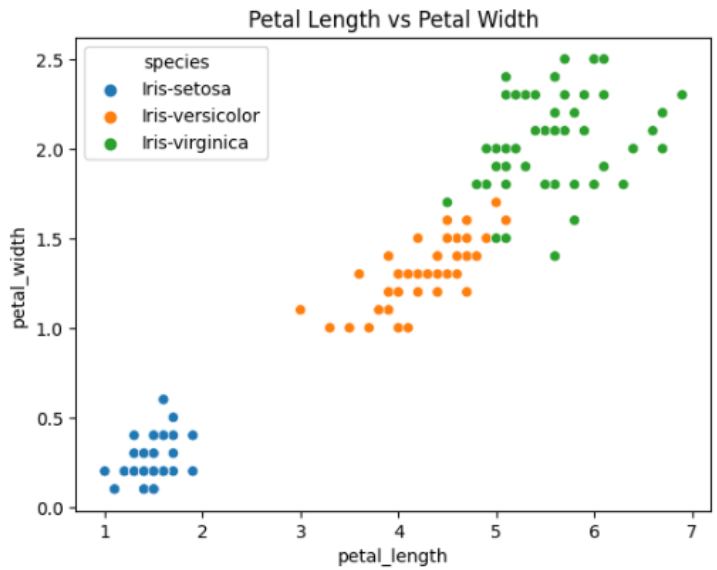


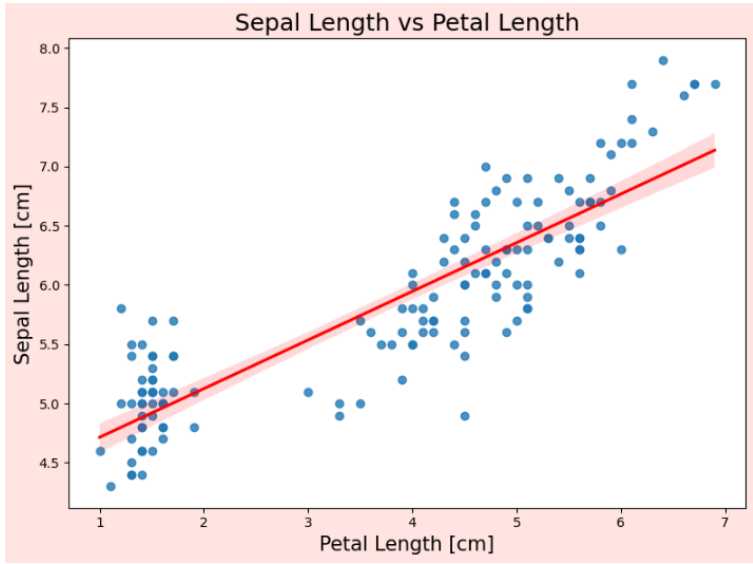
Graficele de tip violin combină aspecte ale diagramelor box și ale diagramelor KDE pentru a oferi mai multe informații despre distribuția datelor. Box plots - Box plots arată distribuția datelor și evidențiază prezența oricăror valori extreme, în timp ce KDE - Kernel Density Estimation arată distribuția punctelor de date pentru fiecare caracteristică (seamănă cu o funcție Gaussiană). Link pentru a înțelege cum să interpretezi graficele de tip violin: https://www.labxchange.org/library/items/lb:LabXchange:46f64d7a:html:1


species
Iris-setosa
Iris-versicolor
Iris-virginica
sepal_length
sepal_width
petal_length
petal_width


Histogram of Petal Length
Frequency
Petal Length


Sepal Length vs Sepal Width
species
Iris-setosa
Iris-versicolor
Iris-virginica
sepal_width
sepal_length


Petal Length vs Petal Width
species
Iris-setosa
Iris-versicolor
Iris-virginica
petal_width
petal_length


Sepal Length vs Petal Length
Sepal Length [cm]
Petal Length [cm]

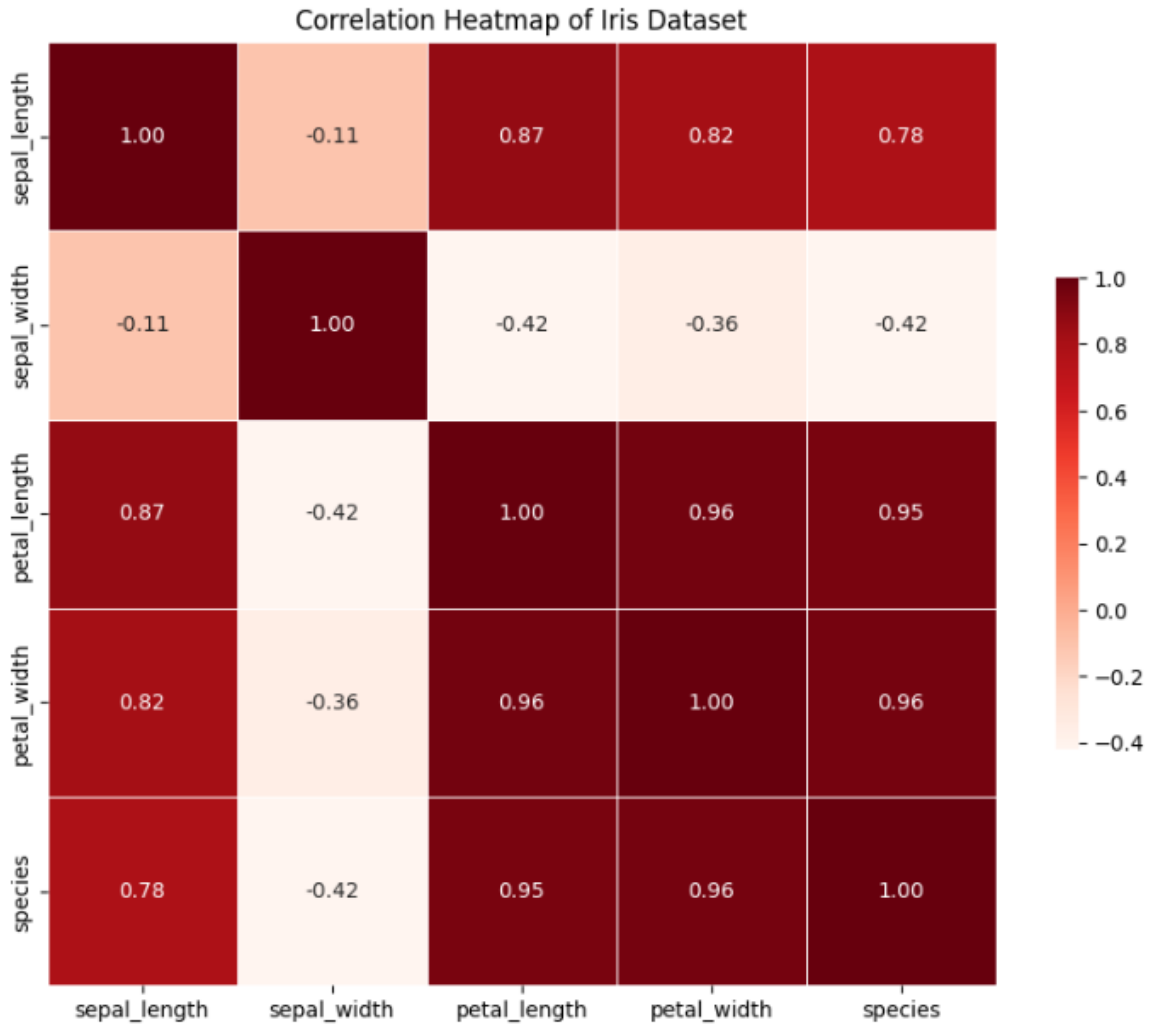
Analiza Coeficientului de Corelație

```
iris.corr() #calc corell matrix
```

	sepal_length	sepal_width	petal_length	petal_width	species
sepal_length	1.000000	-0.109321	0.871305	0.817058	0.782904
sepal_width	-0.109321	1.000000	-0.421057	-0.356376	-0.418348
petal_length	0.871305	-0.421057	1.000000	0.961883	0.948339
petal_width	0.817058	-0.356376	0.961883	1.000000	0.955693
species	0.782904	-0.418348	0.948339	0.955693	1.000000

Observație

Lungimea petalelor și lățimea petalelor au o dependență mai puternică de specie în comparație cu lungimea sepalelor și lățimea sepalelor



Prezentarea teoretică a modelului folosit

Acest model face predicții asupra valorii unei variabile țintă (de exemplu, tipul de floare) prin învățarea unor reguli decizionale simple deduse din datele caracteristice, cum ar fi lungimea și lățimea petalelor și sepalei.

Procesul începe cu **Nodul Rădăcină**, care reprezintă întregul set de date și servește ca primul punct de decizie. Datele sunt apoi supuse unei **Divizări (Splitting)**, unde sunt împărțite în subseturi pe baza unei caracteristici care oferă cea mai bună separare (de exemplu, lungimea petalei). În continuare, se formează **Noduri de Decizie**, care sunt punctele din arbore unde se iau decizii suplimentare pentru a împărți datele în continuare. Procesul continuă până când se ajunge la **Noduri Frunză**, care sunt nodurile finale și conțin clasele de ieșire (de exemplu, Iris Setosa, Iris Versicolor, Iris Virginica).

Decision Tree Classifier

Caracteristici Comune

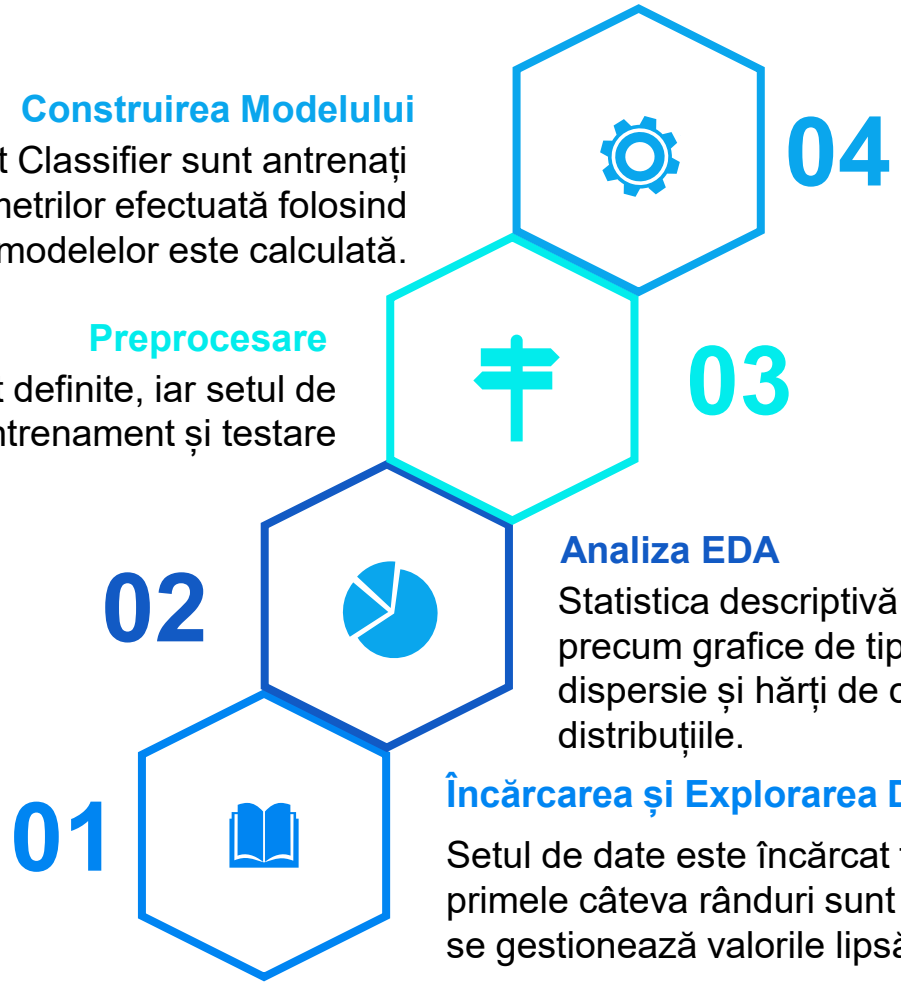
- Învățare supervizată
- Structură bazată pe arbori
- Modele non-liniare

Random Forest Classifier

The Random Forest este o metodă de ansamblu care construiește mai mulți arbori de decizie și combină rezultatele lor pentru a produce predicții mai precise și stabile. Această tehnică reduce supraînvățarea și îmbunătățește acuratețea comparativ cu utilizarea unui singur arbore de decizie prin combinarea rezultatelor din mai mulți arbori și utilizarea votului majoritar pentru predicția finală.

Acest model funcționează printr-un proces numit Bootstrap Aggregating (Bagging), în care subseturi aleatorii ale setului de date sunt utilizate pentru a antrena fiecare arbore de decizie. La fiecare nod al acestor arbori, un subset aleatoriu de caracteristici este selectat pentru a determina cea mai bună divizare, sporind robustețea modelului. Predicția finală este dată de votul majoritar pentru sarcinile de clasificare și media pentru sarcinile de regresie, asigurând rezultate mai precise și stabile prin atenuarea efectelor supraînvățării(overfitting).

Descrierea Implementării



Construirea Modelului

Un Decision Tree Classifier și Random Forest Classifier sunt antrenați și evaluați, cu reglarea hiperparametrilor efectuată folosind GridSearchCV. În final, acuratețea modelelor este calculată.

Preprocesare

Caracteristicile și variabilele țintă sunt definite, iar setul de date este împărțit în seturi de antrenament și testare

Analiza EDA

Statistica descriptivă este calculată folosind describe(), iar vizualizări precum grafice de tip plăcintă, grafice de tip violin, grafice de dispersie și hărți de căldură sunt create pentru a explora relațiile și distribuțiile.

Încărcarea și Explorarea Datelor

Setul de date este încărcat folosind pd.read_csv(), iar forma și primele câteva rânduri sunt inspectate în timp ce se verifică și se gestionează valorile lipsă și intrările duplicate.

Prezentarea rezultatelor + metrici de performanță

Decision Tree Classifier

```
DecisionTreeClassifier()
```

Se calculează acuratețea pe setul de antrenament și pe setul de testare.

```
#Predict and calculate accuracy on the training set
y_train_tree_predicted = decision_model.predict(X_train)
acc1 = accuracy_score(y_train, y_train_tree_predicted)
acc1
```

```
1.0
```

```
#Predict and calculate accuracy on the other set (test)
y_test_tree_predicted = decision_model.predict(X_test)
acc2 = accuracy_score(y_test, y_test_tree_predicted)
acc2
```

```
0.9333333333333333
```

Random Forest Classifier

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [1, 2, 3, 4, 5],
                         'max_features': ['sqrt', 'log2'],
                         'min_samples_split': [3, 5, 7, 9],
                         'n_estimators': [20, 40, 60, 80, 100]},
             scoring='accuracy')
```

estimator: RandomForestClassifier

```
RandomForestClassifier()
```

```
train_accuracy=accuracy_score(train_pred,z_train)
print("Training Accuracy- ",train_accuracy.round(2))
```

```
Training Accuracy-  0.97
```

Prezentarea rezultatelor + metrici de performanță

```
#Predict and calculate accuracy on the training set
y_train_tree_predicted = decision_model.predict(X_train)
acc1 = accuracy_score(y_train, y_train_tree_predicted)
acc1
```

```
1.0
```

```
#Predict and calculate accuracy on the other set (test)
y_test_tree_predicted = decision_model.predict(X_test)
acc2 = accuracy_score(y_test, y_test_tree_predicted)
print("Test Accuracy:", acc2)
# Output predicted species
print("Predicted Species for Test Data:")
for i, pred in enumerate(y_test_tree_predicted):
    print(f"Sample {i + 1}: Predicted Species - {pred}, Actual Species - {y_test.iloc[i]}")
```

```
Test Accuracy: 0.9333333333333333
Predicted Species for Test Data:
Sample 1: Predicted Species - 1, Actual Species - 1
Sample 2: Predicted Species - 2, Actual Species - 2
Sample 3: Predicted Species - 2, Actual Species - 2
Sample 4: Predicted Species - 3, Actual Species - 3
Sample 5: Predicted Species - 1, Actual Species - 1
Sample 6: Predicted Species - 2, Actual Species - 2
Sample 7: Predicted Species - 3, Actual Species - 3
Sample 8: Predicted Species - 1, Actual Species - 1
Sample 9: Predicted Species - 3, Actual Species - 3
Sample 10: Predicted Species - 2, Actual Species - 2
Sample 11: Predicted Species - 1, Actual Species - 1
Sample 12: Predicted Species - 3, Actual Species - 3
Sample 13: Predicted Species - 2, Actual Species - 2
Sample 14: Predicted Species - 2, Actual Species - 3
Sample 15: Predicted Species - 1, Actual Species - 1
```

Tree Classifier

Am adăugat și rezultatele prezise.

```
train_accuracy = accuracy_score(train_pred, z_train)
test_accuracy = accuracy_score(test_pred, z_test)
print("Training Accuracy: ", round(train_accuracy, 2))
print("Test Accuracy: ", round(test_accuracy, 2))

print("Predicted Species for Test Data:")
for i, pred in enumerate(test_pred):
    if i < len(y_test):  # Ensure index is within bounds
        print(f"Sample {i + 1}: Predicted Species - {pred}, Actual Species - {y_test.iloc[i]}")
```

```
Training Accuracy:  0.99
Test Accuracy:  0.96
Predicted Species for Test Data:
Sample 1: Predicted Species - 3, Actual Species - 1
Sample 2: Predicted Species - 2, Actual Species - 2
Sample 3: Predicted Species - 3, Actual Species - 2
Sample 4: Predicted Species - 1, Actual Species - 3
Sample 5: Predicted Species - 2, Actual Species - 1
Sample 6: Predicted Species - 1, Actual Species - 2
Sample 7: Predicted Species - 2, Actual Species - 3
Sample 8: Predicted Species - 1, Actual Species - 1
Sample 9: Predicted Species - 3, Actual Species - 3
Sample 10: Predicted Species - 1, Actual Species - 2
Sample 11: Predicted Species - 1, Actual Species - 1
Sample 12: Predicted Species - 2, Actual Species - 3
Sample 13: Predicted Species - 3, Actual Species - 2
Sample 14: Predicted Species - 2, Actual Species - 3
Sample 15: Predicted Species - 3, Actual Species - 1
```

Random Forest Classifier

Concluzii și observații finale

Concluzii

Dintre modelele utilizate, Random Forest Classifier a arătat o îmbunătățire ușoară față de Clasificatorul de Tip Arbore de Decizie. Această îmbunătățire se datorează în principal capacității metodei de ansamblu de a reduce supraînvățarea, rezultând predicții mai stabile și mai fiabile.

Observații

EDA s-a dovedit crucială în înțelegerea tiparelor și relațiilor care stau la baza setului de date, facilitând o performanță mai bună a modelului. În plus, reglarea hiperparametrului, în special folosind tehnici precum GridSearchCV, a fost esențială pentru optimizarea modelelor, asigurându-se că acestea au fost reglate fin pentru a oferi cea mai bună precizie posibilă. În viitor, analiza ar putea fi extinsă la alte seturi de date pentru a valida robustețea modelelor.

Mulțumesc pentru atenție!

Email: florentina.vera26@yahoo.com