Laboratory 4

Variant #3

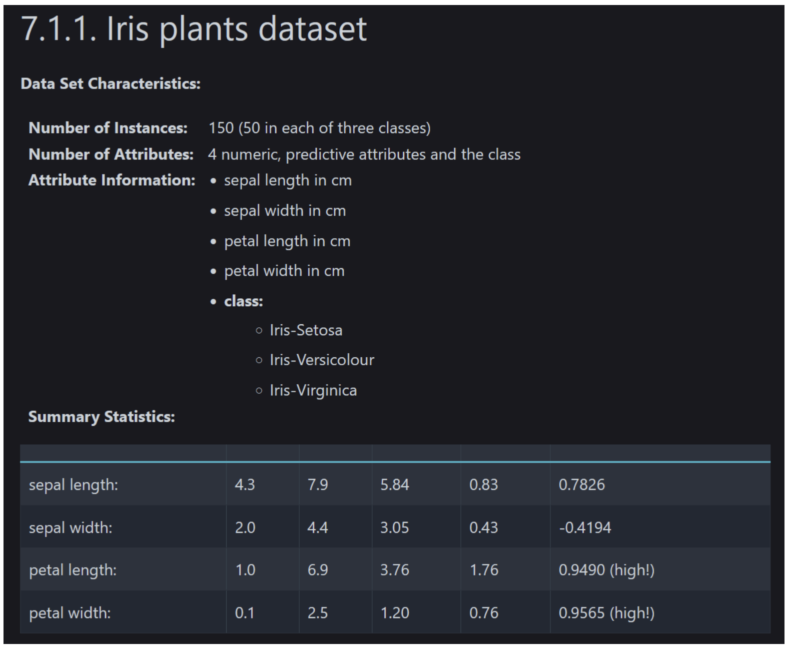
Mariya Zacharneva and Ruslan Melnyk

The Task

The task was to try two models for classifying the dataset of iris flowers.

Dataset

The iris dataset is provided as a toy dataset in the sklearn docs:



7.1.1. Iris plants dataset

Data Set Characteristics:

Number of Instances: 150 (50 in each of three classes)
Number of Attributes: 4 numeric, predictive attributes and the class
Attribute Information:
- sepal length in cm
- sepal width in cm
- petal length in cm
- petal width in cm
- **class:**
  - Iris-Setosa
  - Iris-Versicolour
  - Iris-Virginica

Summary Statistics:

sepal length:	4.3	7.9	5.84	0.83	0.7826
sepal width:	2.0	4.4	3.05	0.43	-0.4194
petal length:	1.0	6.9	3.76	1.76	0.9490 (high!)
petal width:	0.1	2.5	1.20	0.76	0.9565 (high!)

It contains three classes of iris plants, 50 instances for each class. For each flower, we have four numeric attributes: sepal length, sepal width, petal length, and petal width. For petal attributes, we can see the “(high!)” note indicating that it has a very high correlation with the class label, meaning they are excellent predictors. The spread (standard deviation) is higher for petal length, indicating more variability in petal size among flowers.

Models

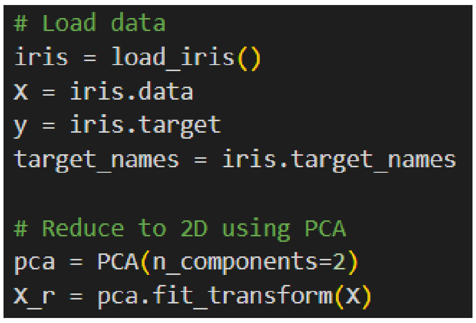
We chose **logistic regression** and **random forest classification** as models because they are commonly used in classification tasks, and we compared them.

Logistic regression is a simple and easy-to-understand model that works well when the data is linearly separable or close to it. It is also fast and gives good results for many basic problems. Random forest is a more advanced model based on decision trees, which is capable of capturing non-linear relationships, especially when the data is more complex. By using both models, we wanted to compare a simple model and a more flexible model, and see which one works better for the task.

While logistic regression assumes that the data is (at least approximately) linearly separable, random forest does not. Logistic regression finds a single straight line/hyperplane, while Random Forest builds complex, piecewise splits in data.

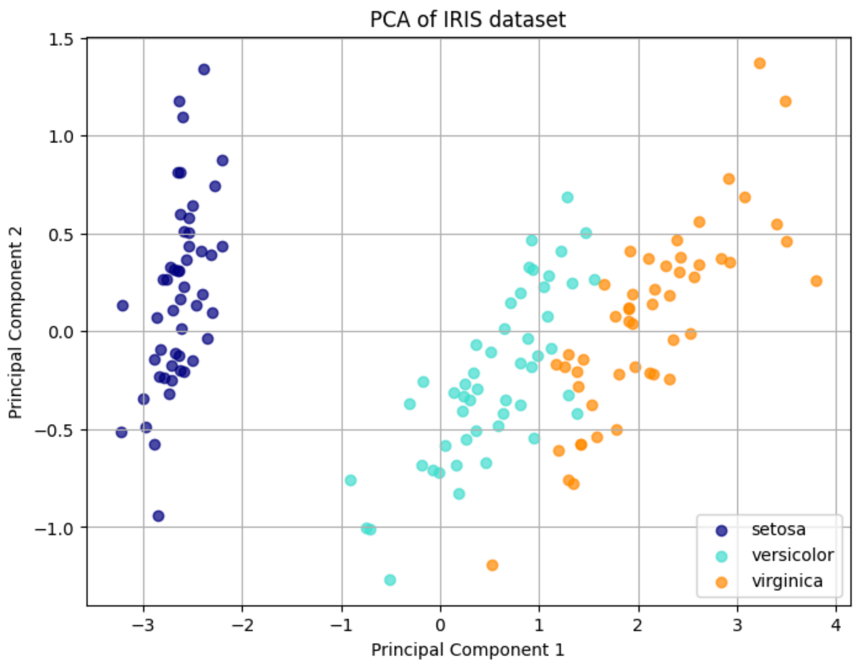
Data analysis

First of all, we can check if the data is in fact linearly separable. Since we have 4 different parameters (meaning 4 dimensions), we cannot simply draw a 2D graph. In this case, we can apply PCA (Principal Component Analysis). This is a dimensionality reduction technique that helps simplify complex datasets by transforming them into fewer dimensions, in the same time keeping as much of the original information (variance) as possible. We can import PCA from `sklearn.decomposition` and use it out of the box:



```
# Load data
iris = load_iris()
X = iris.data
y = iris.target
target_names = iris.target_names

# Reduce to 2D using PCA
pca = PCA(n_components=2)
X_r = pca.fit_transform(X)
```



The Setosa class is clearly linearly separated from the other two classes, when the Versicolor and Virginica classes overlap somewhat, which suggests that these two are not perfectly linearly separable. Logistic regression may work very well for distinguishing Setosa, but might have more difficulty with Versicolor and Virginica due to their overlapping features. However, the overlap is not dramatic, so we expect both model to show good results.

Data Preprocessing

Since the logistic regression method assumes the features are on a similar scale, we are applying StandardScaler to the data: it standardizes features by removing the mean and scaling to unit variance. In other words, it transforms each feature such that it looks like standard normally distributed data (e.g., Gaussian with 0 mean and unit variance).

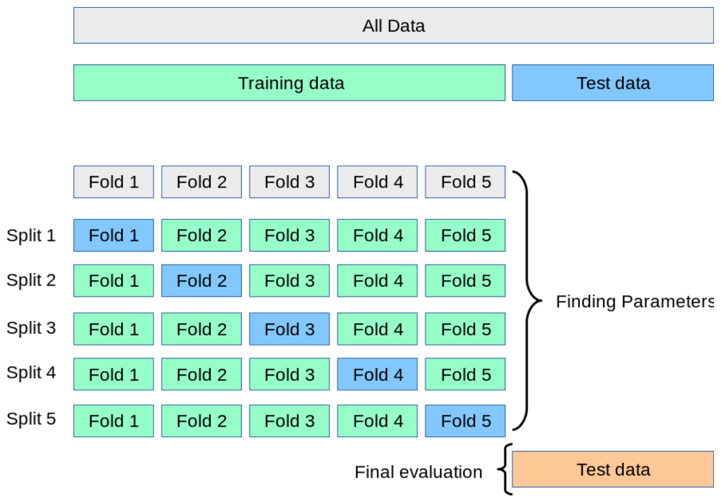
Random forest is scale-invariant, but it doesn't hurt to use a standard scaler, so we will use standartized data here as well.

Data split

Two techniques are used to avoid overfitting:

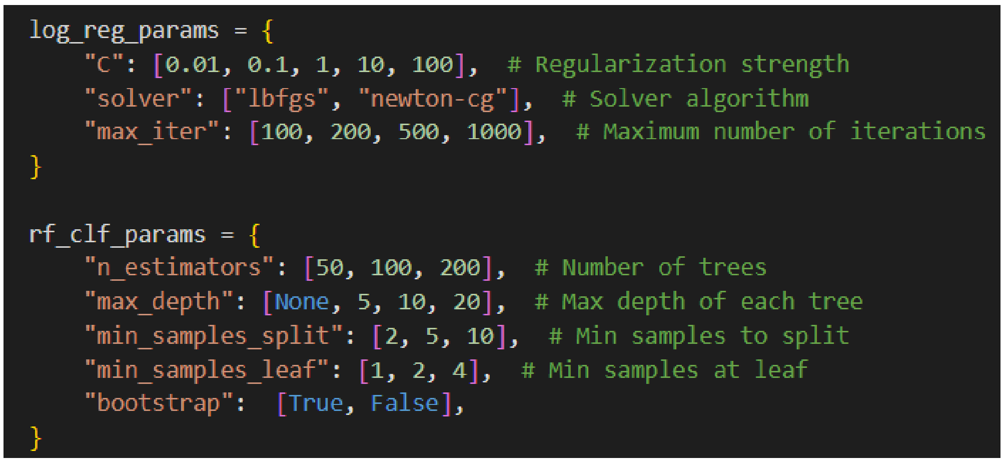
1) We created an 80/20 split of data into a training and a testing subset - the holdout cross-validation approach.

2) To check the performance of the models with different parameters, we use the 4-fold cross-validation: this is a technique that allows us to evaluate the model more than once by splitting the testing subset into several splits and running the model on each of them separately.



Parameters

We have decided to try the following parameters for models:



```
log_reg_params = {
    "C": [0.01, 0.1, 1, 10, 100],  # Regularization strength
    "solver": ["lbfgs", "newton-cg"],  # Solver algorithm
    "max_iter": [100, 200, 500, 1000],  # Maximum number of iterations
}

rf_clf_params = {
    "n_estimators": [50, 100, 200],  # Number of trees
    "max_depth": [None, 5, 10, 20],  # Max depth of each tree
    "min_samples_split": [2, 5, 10],  # Min samples to split
    "min_samples_leaf": [1, 2, 4],  # Min samples at leaf
    "bootstrap":  [True, False],
}
```

Quick overview of parameters for logistic regression:

- **C**: regularization strength, it controls how strongly the model tries to avoid overfitting; a smaller value means stronger regularization.
- **solver**: the algorithm to use in the optimization problem; here we are testing 3 algorithms:
    - “lbfgs” is the best general-purpose solver for multiclass problems.
    - “newton-cg” is also good for multiclass problems and can be more stable in some cases, but is in general slower.
- **max_iter**: maximum number of iterations the solver runs; too high value slows training, when too low can result in undertraining of the model.

Parameters for random forest:

- **n_estimators**: the number of decision trees in the model.
- **max_depth**: maximum depth of each decision tree; when not limited, the model can overfit.
- **min_samples_split**: minimum number of samples needed to split a node; 5 or 10 forces trees to be more conservative, often improves generalization.
- **min_samples_leaf**: minimum number of samples required to be at a leaf node; 1 can result in very specific, overfitted rules, when 2 or 4 encourage more generalization by preventing tiny leaf nodes.
- **bootstrap**: whether bootstrap samples are used when building trees; if false, the whole dataset is used to build each tree.

We utilized grid search with cross-validation to pick the final model showing the best mean accuracy over the 4 folds. The grid search will try all possible combinations of parameters and check which one shows the best performance.

Results

You can find full results for all parameter combinations in **`lab_4_full_results.pdf`**.

Logistic Regression

Grid search with cross-validation indicated:

> Best Logistic Regression Accuracy: 0.97
> Best Logistic Regression Params: {'C': 100, 'max_iter': 100, 'solver': 'lbfgs'}

The high value of C indicates that less regularization was optimal in this case, likely because the data is clean and relatively easy to separate. Notice that it picks first best, but the same result is achieved with different parameter combinations as well (see **`lab_4_full_results.pdf`**). In general, logistic regression shows extremely good results in this classification task.

Random Forest

Grid search with cross-validation indicated:

```
Best Random Forest Accuracy: 0.96
Best Random Forest Params: {'bootstrap': True, 'max_depth': None,
                            'min_samples_leaf': 1, 'min_samples_split': 10,
                            'n_estimators': 100}
```

The deeper trees and larger minimum split size helped the model avoid overfitting and have enough flexibility to capture slightly non-linear relationships.

Comparison

Eventually, we fit the final models on the whole dataset (remember that we used k-fold cross-validation before) and compare the performance of both models.

Logistic Regression Classification Report:

```
              precision    recall    f1-score    support
         0    1.00         1.00      1.00        12
         1    1.00         1.00      1.00        8
         2    1.00         1.00      1.00        10

accuracy                             1.00        30
macro avg     1.00         1.00      1.00        30
weighted avg  1.00         1.00      1.00        30
```

Random Forest Classification Report:

```
              precision    recall    f1-score    support
         0    1.00         1.00      1.00        12
         1    1.00         1.00      1.00        8
         2    1.00         1.00      1.00        10

accuracy                             1.00        30
macro avg     1.00         1.00      1.00        30
weighted avg  1.00         1.00      1.00        30
```

Conclusion:

We can see that both models showed 100% accuracy on the unseen 20% of the test subset in accuracy, precision, recall, and F1-score across all classes. This outcome confirms that both models are highly effective for this classification task and are able to fully distinguish between the three iris species. In conclusion, either model is suitable for this dataset, but Logistic Regression may be preferred due to its efficiency and slightly better validation performance.