Atelier 4 : Ensemble Learning (Bagging)

Pr. Ali Idri

Master of Science in Quantitative and Financial Modelling

2022-2023

Problem statement:

The Pima Indians dataset is a classic dataset in machine learning and is often used for classification tasks. It contains data on Pima Indian women aged 21 and older living near Phoenix, Arizona, USA. The goal of the classification task is to predict whether or not a woman will develop diabetes based on various medical measurements.

This task of classification is at the heart of medical diagnostic decision-support systems, which are becoming more widely used in the healthcare industry. Such machine-learning algorithms can greatly reduce the false negative rate (when patients are misdiagnosed until it is too late) and improve treatment and survival chances. It remains a difficult task due to subtle variations in patient data, making it hard for a **single model** to achieve **high performance**. Effective models can be constructed using ensemble learning that can improve the accuracy of the classification task by reducing overfitting, increasing the diversity of the models, and improving the generalization performance of the model.

Bagging is an ensemble learning method that involves training multiple models on different subsets of the training data and then combining their predictions. One popular algorithm that uses bagging is the Random Forest algorithm. In this case, one can train multiple decision tree models on different subsets of the training data and combine their predictions using voting or averaging.

This workshop focusses on Bagging and tries to answer four research questions:

1- Do the bagging ensembles outperform their base learners?
2- What is the number of estimators that gives the best results for each bagging ensemble?
3- What is the best performing bagging ensemble?
4- Do the bagging ensembles outperform the single classifiers trained on all the training dataset?

This case study will be tackled using:

- ✓ The scikit-learn library
- ✓ The bagging ensemble method for classification
- ✓ Single machine learning classifiers

The examples of source code presented in this tutorial are proposals. At the end of the tutorial, you can make your own implementations and interpretations.

I. Loading and pre-processing the Data:

-Let's start by importing the libraries:

```
# Import libraries
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import pandas as pd
```

We load the Pima Indians dataset and we split the data into dependent and independent variables:

```
# Load the Pima Indians dataset
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
feature_names = names[:8]
data = pd.read_csv(url, names=names)

#Splitting the data into dependent and independent variables
X = data.drop("class", axis=1)
y=  data["class"]
```

II. Training the single classifiers:

We will train and evaluate scikit-learn's homogeneous parallel ensemble modules in practice using different base learners: Decision Tree (DT), Multi-Layer Perceptron (MLP) and K-nearest neighbors (KNN) and test different number of estimators.

1) Decision Tree

Let's start by training the DT classifier using data split with a ration of 70% for training and 30% for testing.

```
# Split the data into training and testing sets
X1, X2, y1, y2 = train_test_split(X,y, random_state= 42, train_size=0.7)

# The DT classifier
DT = DecisionTreeClassifier()
DT.fit(X1,y1).predict(X2)
DT_Pred=DT.predict(X2)
DT_results = accuracy_score(y2,DT_Pred)
print(accuracy_score(y2,DT_Pred))
```

From the results, we notice that when using the DT classifier with its default configuration given with the SKLEARN library, we obtain ana accuracy value of 70.56%

2) KNN

We keep the same data splitting ratio for the KNN classifier

```
# The KNN classifier
KNN = KNeighborsClassifier()
KNN.fit(X1,y1).predict(X2)
KNN_Pred=KNN.predict(X2)
KNN_results = accuracy_score(y2,KNN_Pred)
print(accuracy_score(y2,KNN_Pred))
```

From the results, we notice that when using the KNN classifier with its default configuration given with the SKLEARN library, we obtain ana accuracy value of 68.83%

3) MLP

Same thing for the MLP classifier with number of iteration = 250 and batch size set to 32:

```
# The MLP classifier
MLP = MLPClassifier(batch_size=32,max_iter=250)
MLP.fit(X1,y1).predict(X2)
MLP_Pred=MLP.predict(X2)
MLP_results = accuracy_score(y2,MLP_Pred)
print(accuracy_score(y2,MLP_Pred))
```

From the results, we notice that when using the MLP classifier with a batch size of 32 and 250 iterations, we obtain ana accuracy value of 69.26%

Question 1: Train an SVM classifier, what is the performance of SVM on the same dataset?

III. Training the Bagging ensembles:

In this section, we will train variants of the bagging classifier by changing the base learner (estimator): DT, KNN and MLP, and number of estimators (3, 7, 10, 15 and 20).

1) Bagging using DT as base learner (Random Forest) with 3, 7, 10, 15 and 20 estimators:

For more details on the parameters of the Sklearn.Ensemble.BaggingClassifier, you can visit the page: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html?highlight=bagging%20classifier#sklearn.ensemble.BaggingClassifier

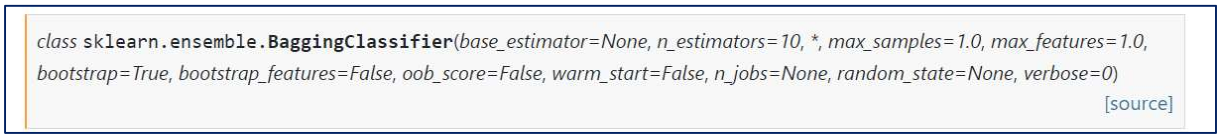

Fig 1: Bagging Classifier on the SKLEARN.ENSEMBLE library

Let's start by implementing the bagging using the DT classifier with different number of estimators:

```
#3 estimators
DBG3 = BaggingClassifier(estimator=DecisionTreeClassifier(),max_samples=0.6, n_estimators=3,
random_state=42)
DBG3.fit(X1,y1).predict(X2)
DBG3_Pred=DBG3.predict(X2)
DBG3_results = accuracy_score(y2,DBG3_Pred)
print(accuracy_score(y2,DBG3_Pred))

#7 estimators
DBG7 = BaggingClassifier(estimator=DecisionTreeClassifier(),max_samples=0.6, n_estimators=7,
random_state=42)
DBG7.fit(X1,y1).predict(X2)
DBG7_Pred=DBG7.predict(X2)
DBG7_results = accuracy_score(y2,DBG7_Pred)
print(accuracy_score(y2,DBG7_Pred))

#10 estimators
DBG10 = BaggingClassifier(estimator=DecisionTreeClassifier(),max_samples=0.6, n_estimators=1
random_state=42)
DBG10.fit(X1,y1).predict(X2)
DBG10_Pred=DBG10.predict(X2)
DBG10_results = accuracy_score(y2,DBG10_Pred)
print(accuracy_score(y2,DBG10_Pred))


#15 estimators
DBG15 = BaggingClassifier(estimator=DecisionTreeClassifier(),max_samples=0.6, n_estimators=1
random_state=42)
DBG15.fit(X1,y1).predict(X2)
DBG15_Pred=DBG15.predict(X2)
DBG15_results = accuracy_score(y2,DBG15_Pred)
print(accuracy_score(y2,DBG15_Pred))
```

```
#20 estimators
DBG20 = BaggingClassifier(estimator=DecisionTreeClassifier(),max_samples=0.6, n_estimators=2
random_state=42)
DBG20.fit(X1,y1).predict(X2)
DBG20_Pred=DBG20.predict(X2)
DBG20_results = accuracy_score(y2,DBG20_Pred)
print(accuracy_score(y2,DBG20_Pred))
```

And then plot the results of the three developed bagging ensembles using DT as base learner and the number of estimators: 3, 7, 10, 15 and 20.

```
#We plot the  accuracy of the different bagging ensembles using three estimators
models = ['DBG3','DBG7','DBG10','DBG15','DBG20']
n_estimators = [3,7,10,15,20]
scores=[DBG3_results*100,DBG7_results*100, DBG10_results*100,
DBG15_results*100,DBG20_results*100]

plt.plot(n_estimators, scores, 'ro-',)
plt.xlabel("Number of estimators")
plt.ylabel("Accuracy (%)")
plt.show()
```

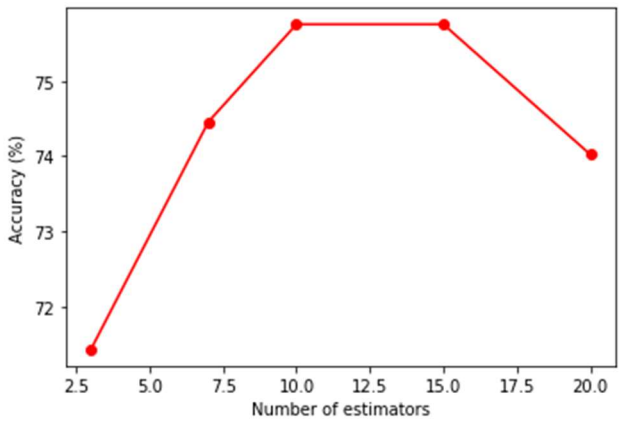

Fig 2: Plot of the Bagging Classifier using DT with different number of estimators

-How does the performance of the DT bagging ensemble change when we add more estimators?

2) Bagging using KNN as base learner with 3, 7, 10, 15 and 20 estimators:

Implementation of the bagging classifier using the KNN classifier with different number of estimators:

```
#3 estimators
KBG3 = BaggingClassifier(estimator=KNeighborsClassifier(),max_samples=0.6, n_estimators=3,
random_state=42)
KBG3.fit(X1,y1).predict(X2)
```

```
KBG3_Pred=KBG3.predict(X2)
KBG3_results = accuracy_score(y2,KBG3_Pred)
print(accuracy_score(y2,KBG3_Pred))

#7 estimators
KBG7 = BaggingClassifier(estimator=KNeighborsClassifier(),max_samples=0.6, n_estimators=7,
random_state=42)
KBG7.fit(X1,y1).predict(X2)
KBG7_Pred=KBG7.predict(X2)
KBG7_results = accuracy_score(y2,KBG7_Pred)
print(accuracy_score(y2,KBG7_Pred))

#10 estimators
KBG10 = BaggingClassifier(estimator=KNeighborsClassifier(),max_samples=0.6, n_estimators=10,
random_state=42)
KBG10.fit(X1,y1).predict(X2)
KBG10_Pred=KBG10.predict(X2)
KBG10_results = accuracy_score(y2,KBG10_Pred)
print(accuracy_score(y2,KBG10_Pred))


#15 estimators
KBG15 = BaggingClassifier(estimator=KNeighborsClassifier(),max_samples=0.6, n_estimators=15,
random_state=42)
KBG15.fit(X1,y1).predict(X2)
KBG15_Pred=KBG15.predict(X2)
KBG15_results = accuracy_score(y2,KBG15_Pred)
print(accuracy_score(y2,KBG15_Pred))

#20 estimators
KBG20 = BaggingClassifier(estimator=KNeighborsClassifier(),max_samples=0.6, n_estimators=20,
random_state=42)
KBG20.fit(X1,y1).predict(X2)
KBG20_Pred=KBG20.predict(X2)
KBG20_results = accuracy_score(y2,KBG20_Pred)
print(accuracy_score(y2,KBG20_Pred))
```

Then, we plot the results of the three developed bagging ensembles using KNN as base learner and the number of estimators: 3, 7, 10, 15 and 20.

```
#We plot the  accuracy of the different bagging ensembles using three estimators
models = ['KBG3','KBG7','KBG10','KBG15','KBG20']
n_estimators = [3,7,10,15,20]
scores=[KBG3_results*100,KBG7_results*100, KBG10_results*100,
KBG15_results*100,KBG20_results*100]
```

```
plt.plot(n_estimators, scores, 'ro-',)
plt.xlabel("Number of estimators")
plt.ylabel("Accuracy (%)")
plt.show()
```

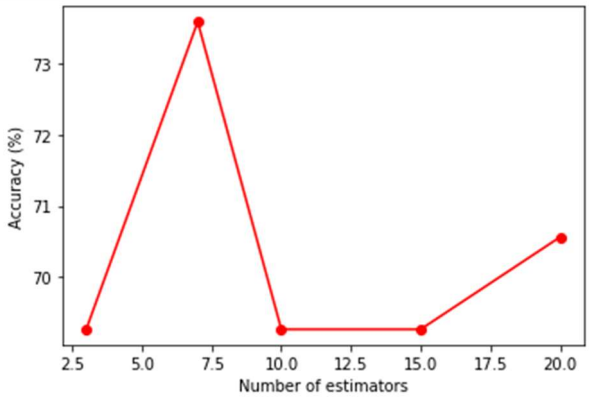

Fig 3: Plot of the Bagging Classifier using KNN with different number of estimators

- How does the performance of the KNN bagging ensemble change when we add more estimators?

3) Bagging using MLP as base learner with 3, 7, 10, 15 and 20 estimators:

Implementation of the bagging classifier using the MLP classifier with different number of estimators:

```
#3 estimators
MBG3 = BaggingClassifier(estimator=MLPClassifier(batch_size=32,max_iter=250),max_samples=0.
n_estimators=3, random_state=42)
MBG3.fit(X1,y1).predict(X2)
MBG3_Pred=MBG3.predict(X2)
MBG3_results = accuracy_score(y2,MBG3_Pred)
print(accuracy_score(y2,MBG3_Pred))

#7 estimators
MBG7 = BaggingClassifier(estimator=MLPClassifier(batch_size=32,max_iter=250),max_samples=0.
n_estimators=7, random_state=42)
MBG7.fit(X1,y1).predict(X2)
MBG7_Pred=MBG7.predict(X2)
MBG7_results = accuracy_score(y2,MBG7_Pred)
print(accuracy_score(y2,MBG7_Pred))

#10 estimators
MBG10 = BaggingClassifier(estimator=MLPClassifier(batch_size=32,max_iter=250),max_samples=0.
n_estimators=10, random_state=42)
MBG10.fit(X1,y1).predict(X2)
```

```
MBG10_Pred=MBG10.predict(X2)
MBG10_results = accuracy_score(y2,MBG10_Pred)
print(accuracy_score(y2,MBG10_Pred))


#15 estimators
MBG15 = BaggingClassifier(estimator=MLPClassifier(batch_size=32,max_iter=250),max_samples=0.
n_estimators=15, random_state=42)
MBG15.fit(X1,y1).predict(X2)
MBG15_Pred=MBG15.predict(X2)
MBG15_results = accuracy_score(y2,MBG15_Pred)
print(accuracy_score(y2,MBG15_Pred))

#20 estimators
MBG20 = BaggingClassifier(estimator=MLPClassifier(batch_size=32,max_iter=250),max_samples=0.
n_estimators=20, random_state=42)
MBG20.fit(X1,y1).predict(X2)
MBG20_Pred=MBG20.predict(X2)
MBG20_results = accuracy_score(y2,MBG20_Pred)
print(accuracy_score(y2,MBG20_Pred))
```

Then, we plot the results of the three developed bagging ensembles using MLP as base learner and the number of estimators: 3, 7, 10, 15 and 20.

```
#We plot the  accuracy of the different bagging ensembles using three estimators
models = ['MBG3','MBG7','MBG10','MBG15','MBG20']
n_estimators = [3,7,10,15,20]
scores=[MBG3_results*100,MBG7_results*100, MBG10_results*100,
MBG15_results*100,MBG20_results*100]

plt.plot(n_estimators, scores, 'ro-',)
plt.xlabel("Number of estimators")
plt.ylabel("Accuracy (%)")
plt.show()
```

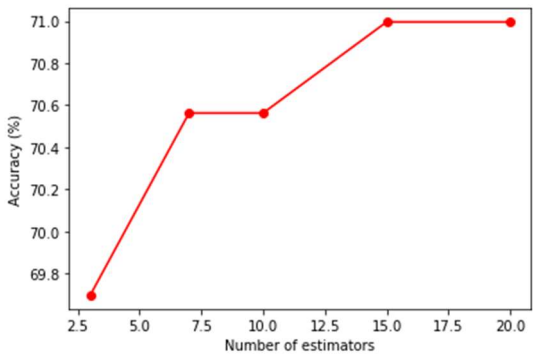

Fig 4: Plot of the Bagging Classifier using MLP with different number of estimators

- How does the performance of the MLP bagging ensemble change when we add more estimators?

Question 2: Construct bagging ensembles with different number of estimators using SVM as base learner, how does the performance of the SVM bagging ensemble change when we add more estimators?

IV. Comparing the bagging ensembles with their base learners:

In this section we compare the bagging ensembles developed using the three classifiers: DT, KNN and MLP with their base learners. To do so we compare as an example the bagging ensemble of 3 estimators with the 3 base learners.

```
r1_dt=accuracy_score(y2,DBG3.estimators_[0].predict(X2))
r2_dt=accuracy_score(y2,DBG3.estimators_[1].predict(X2))
r3_dt=accuracy_score(y2,DBG3.estimators_[2].predict(X2))

r1_knn=accuracy_score(y2,KBG3.estimators_[0].predict(X2))
r2_knn=accuracy_score(y2,KBG3.estimators_[1].predict(X2))
r3_knn=accuracy_score(y2,KBG3.estimators_[2].predict(X2))

r1_mlp=accuracy_score(y2,MBG3.estimators_[0].predict(X2))
r2_mlp=accuracy_score(y2,MBG3.estimators_[1].predict(X2))
r3_mlp=accuracy_score(y2,MBG3.estimators_[2].predict(X2))

plotdata = pd.DataFrame({

   "Bagging ensemble":[DBG3_results*100,KBG3_results*100,MBG3_results*100],
   "Base learner 1":[r1_dt*100,r1_knn*100,r1_mlp*100],
   "Base learner 2":[r2_dt*100,r2_knn*100,r2_mlp*100],
   "Base learner 3":[r3_dt*100,r3_knn*100,r3_mlp*100]},
   index=["DT", "KNN", "MLP"]
 )

plotdata.plot(kind="bar",figsize=(15, 8))
plt.title("Comparaison of the Bagging ensemble of 5 estimators with its base learners")

plt.xlabel("Bagging models")

plt.ylabel("Accuracy (%)")
```

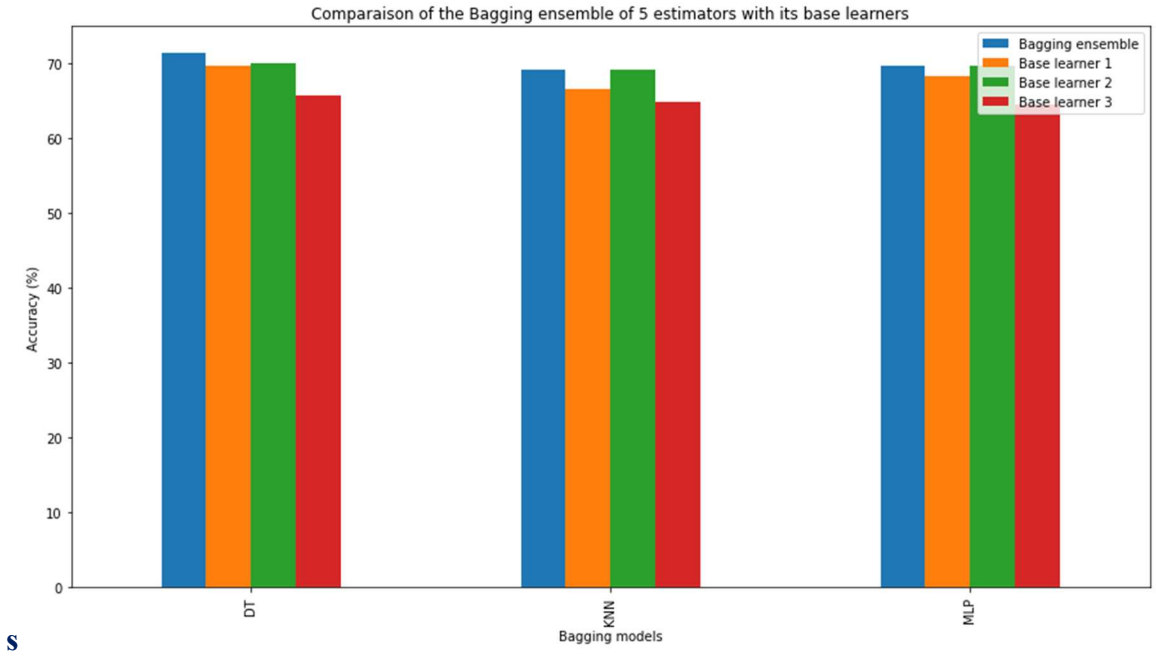

Fig 5: Comparison of the bagging ensembles with their base learners

Figure 5 illustrates the comparison of the bagging ensembles using 3 estimators with their base learners. It is noticeable that:

- For the DT and KNN classifiers, the bagging ensembles outperformed their base learners
- For the MLP classifier, one of the base learner had almost a similar performance with the bagging ensemble.

Question 3: Do the same comparison with the bagging ensembles of 7 estimators and their 7 base learners.

V. Comparing the built bagging ensembles:

In this section we compare the different bagging ensembles developed using the three classifiers: DT, KNN and MLP and five number of estimators: 3, 7, 10, 15 and 20 by plotting the accuracy of the ensembles using different number of trees and classifiers.

```
plotdata = pd.DataFrame({

    "DT":[DBG3_results*100,DBG7_results*100,DBG10_results*100,DBG15_results*100, DBG20_results*100

    "KNN":[KBG3_results*100,KBG7_results*100,KBG10_results*100,KBG15_results*100,
KBG20_results*100],

    "MLP":[MBG3_results*100,MBG7_results*100,MBG10_results*100,MBG15_results*100,
MBG20_results*100]},
```

```
    index=["Bagging of 3 estimators ", "Bagging of 7 estimators","Bagging of 10 estimators", "Bagging of 15 estimators", "Bagging of 20 estimators"])

plotdata.plot(kind="bar",figsize=(15, 8))
plt.title("Comparaison of the Bagging ensembles of the three classifier : DT, KNN and MLP for each number of estimators: 3, 7, 10, 15 and 20")

plt.xlabel("Bagging models")

plt.ylabel("Accuracy (%)")
```

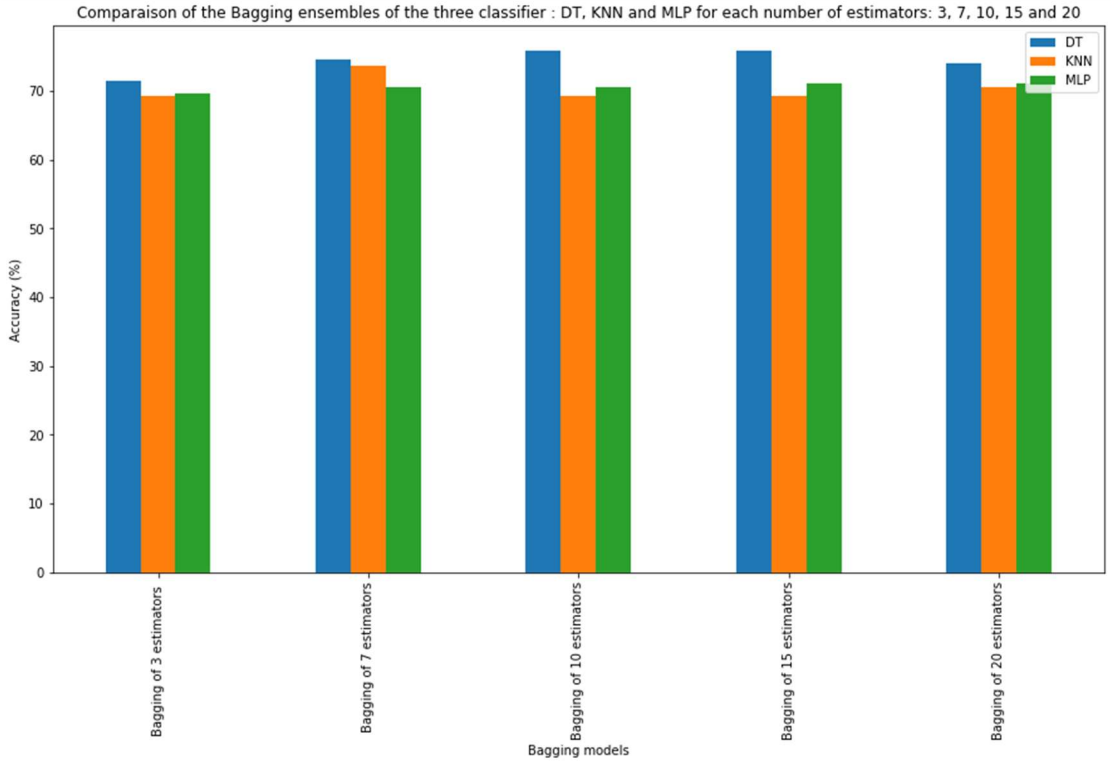

Fig 6: Comparison of the bagging ensembles regardless the classifier

Figure 6 illustrates the comparison of the bagging ensembles using different number of estimators and different classifiers as base learners. It is noticeable that regardless the number of estimators, the bagging ensemble built using the MLP classifier as a base learner outperformed the other bagging ensembles.

Question 4: Add the five SVM bagging ensembles to the previous figure and compare it with the rest of the ensembles.

VI. Comparing the single classifiers and the bagging ensemble:

In this section we compare the bagging ensembles developed using the three classifiers: DT, KNN and MLP and five number of estimators: 3, 7, 10, 15 and 20 with their base learners. To do so we compare each bagging ensemble with its base learner.

```
plotdata = pd.DataFrame({

    "Bagging of 3":[DBG3_results*100,KBG3_results*100,MBG3_results*100],

    "Bagging of 7":[DBG7_results*100,KBG7_results*100,MBG7_results*100],

    "Bagging of 10":[DBG10_results*100,KBG10_results*100,MBG10_results*100],

    "Bagging of 15":[DBG15_results*100,KBG15_results*100,MBG15_results*100],

    "Bagging of 20":[DBG20_results*100,KBG20_results*100,MBG20_results*100],

    "Single classifiers":[DT_results*100,KNN_results*100,MLP_results*100]},

    index=["DT", "KNN", "MLP"])

plotdata.plot(kind="bar",figsize=(20, 10))
plt.title("Comparaison of the Bagging ensembles and the single classifiers")

plt.xlabel("Bagging and base learners models")

plt.ylabel("Accuracy (%)")
```

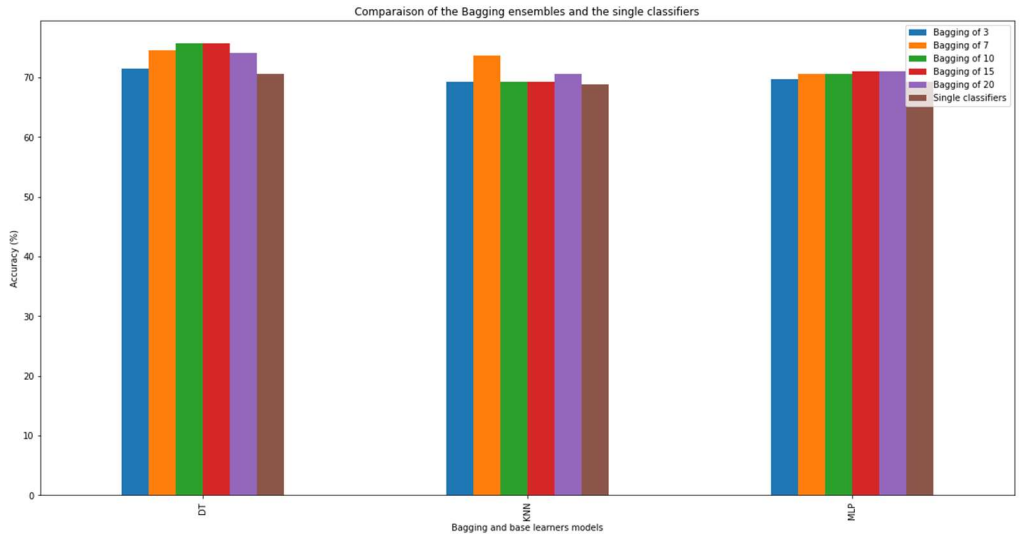

Fig 7: Comparison of the bagging ensembles with the single classifiers

Figure 7 illustrates the comparison of the bagging ensembles using different number of estimators with their base learner.

Question 5: **Add the SVM bagging ensembles to the previous figure and compare it with the SVM single classifier.**