8. Ensemble Learning in ML

1. Introduction

- The Ensemble methods in machine learning combine the insights obtained from multiple learning models to facilitate accurate and improved decisions.

- There are two main types of ensemble learning:
1. Bagging (Bootstrap Aggregating)
2. Boosting.

Ensemble Learning Techniques

Basic Ensemble Techniques:
- Max voting
- Averaging
- Weighted average

Advanced Ensemble Techniques:
- Stacking
- Blending
- Bagging
- Boosting.

Algorithms Based on Bagging:
- Bagging meta-estimator
- Random forest

Algorithms Based on Boosting:
- AdaBoost

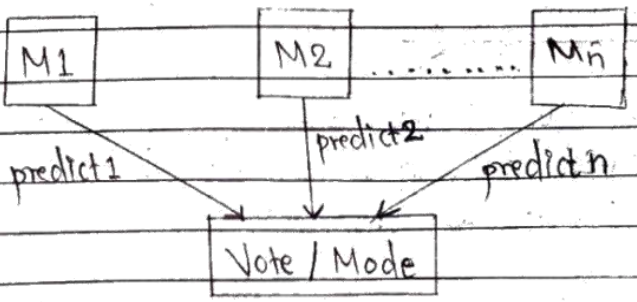
- GBM
- XGBM
- Light GBM
- Cat Boost

2. Basic Ensemble Techniques

1. Max Voting

↳ The max voting method is generally for classification problems. In this technique, multiple models are used to make predictions for each data point.

↳ The predictions by each model are considerd as a 'vote'. The predictions which we get from majority of the models are used as the final prediction.

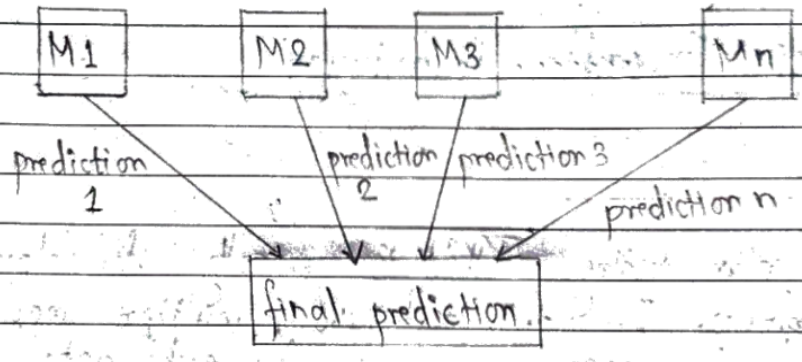


2. Averaging and Weighted Average Voting

↳ Take an average of predictions from all the models

and use it to make the final prediction.

- Averaging can be used for making predictions in regression problems or while calculating probabilities for classification problems.



- Weighted means we have assigned the weight to models.

Working Practically

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Voting Classifier :

```
# Create a dataset using make_moons
from sklearn.datasets import make_moons

# Get data for dataset
```

```
x, y = make_moons(n_samples = 1000, noise = 0.2)
# Here x is independent and y is dependent data

# Create a dataframe using x and y for tabular
# visualization
df = {"x1": x[:,0], "x2": x[:,1], "y": y}
dataset = pd.DataFrame(df)
dataset.head(3)

# Visualize data
sns.scatterplot(x = "x1", y = "x2", data = dataset, hue =
                "y")
plt.show()

# Train test split
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y,
        test_size = 0.2, random_state = 42)

# Train different models
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB

# DecisionTree Classifier
dtc = DecisionTreeClassifier()
dtc.fit(x_train, y_train)
dtc.score(x_train, y_train), dtc.score(x_test, y_test)

# SVC
```

```
svc = SVC()
svc.fit(x_train, y_train)
svc.score(x_train, y_train), svc.score(x_test, y_test)

# Use Ensemble Learning (VotingClassifier)
from sklearn.ensemble import VotingClassifier

# Create a list of tuples with classifier names and
# instances (estimator)
l = [("dtc", DecisionTreeClassifier()),
     ("svc", SVC()),
     ("gnb", GaussianNB())]

# Create the Voting Classifier model.
vc = VotingClassifier(l)
# You can also use weightage in terms of list:
#  weights = [1, 2, 3]
vc.fit(x_train, y_train)

vc.score(x_train, y_train), vc.score(x_test, y_test)

# Lets create a dataframe for clear visualization
predicted = {
    "DecisionTreeClassifier": dtc.predict(x_test),
    "SVC": svc.predict(x_test),
    "GaussianNB": gnb.predict(x_test),
    "VotingClassifier": vc.predict(x_test),
}
# Create a dataframe
pd.DataFrame(predict)
# The most repeated result (mode) is final result
```

by Voting Classifier

Voting Regressor :

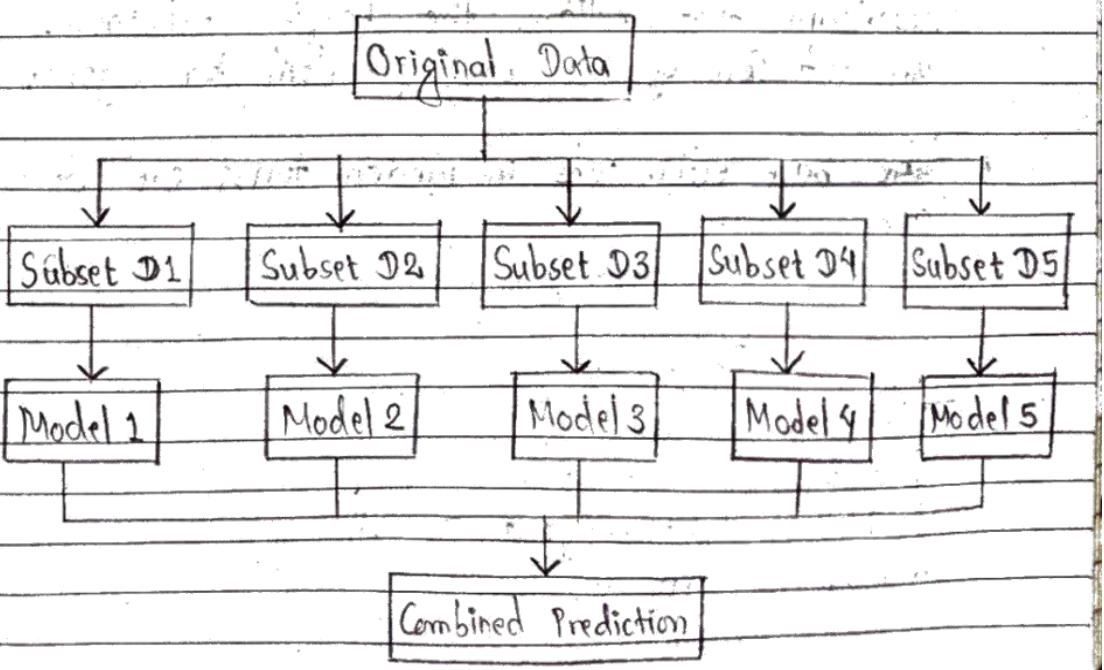
```
dataset = pd.read_csv("Maths.csv")
dataset.head(3)

# Check for null values
dataset.isnull().sum()

# Remove all the null values instead of filling (big
# dataset)
dataset.dropna(inplace=True)

# Select dependent and independent data
x = dataset[["Number1"]]
y = dataset["Total Sum"]

# Train test split
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x,
        y, test_size=0.2, random_state=42)

# Train different models
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR

# LinearRegression
lr = LinearRegression()
```

```
lr.fit(x_train, y_train)
lr.score(x_train, y_train), lr.score(x_test, y_test)

# Decision Tree Regressor
dtr = DecisionTreeRegressor()
dtr.fit(x_train, y_train)
dtr.score(x_train, y_train), dtr.score(x_test, y_test)

# SVR
svr = SVR()
svr.fit(x_train, y_train)
svr.score(x_train, y_train), svr.score(x_test, y_test)

# Use Ensemble Learning (Voting Regressor)
from sklearn.ensemble import VotingRegressor

# Create a list of tuples with classifier names and
# instances (estimators)
l = [("lr", LinearRegression()), ("dtr",
    DecisionTreeRegressor()), ("svr", SVR())]

vr = VotingRegressor(l, weights=[2,3,4])
vr.fit(x_train, y_train)

vr.score(x_train, y_train), vr.score(x_test, y_test)

# Now lets see how VotingRegressor works by
# comparision on dataset.
df = {
    "LinearRegression": lr.predict(x_test),
    "DecisionTreeRegressor": dtr.predict(x_test),
```

```
    "SVR" : svr.predict(x_test)
    "VotingRegressor" : vr.predict(x_test),
}
# Create a dataframe
pd.DataFrame(df).head(3)
# VotingRegressor gives average of other models
```

3. Bagging

- Bagging (or Bootstrap Aggregating) technique uses these subsets (bags) to get a fair idea of the distribution (complete set).
- The size of subsets created for bagging may be less than the original set.



Date
Page 193

↳ Bagging algorithms are:
1. Bagging meta-estimator
2. Random forest

1. Bagging Meta-Estimator

↳ Bagging meta-estimator is an ensembling algorithm that can be used for both classification (Bagging Classifier) and regression (Bagging Regressor) problems.

↳ It follows the typical bagging technique to make predictions.

2. Random Forest

↳ Random forest is another ensemble machine learning algorithm that follows the bagging technique. It is an extension of the bagging estimator algorithm.

↳ The base estimators in random forest are decision trees.

Working Practically

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Classification Analysis:

```
# Create a dataset using make_moons
from sklearn.datasets import make_moons

# Get data for dataset
x, y = make_moons (n_samples = 1000, noise = 0.2)
# Here x is independent and y is dependent data

# Create a dataframe using x and y for tabular
# visualization
df = {"x1": x[:,0], "x2": x[:,1], "y": y}
dataset = pd.DataFrame (df)
dataset.head (3)

# Visualize data
sns.scatterplot (x = "x1", y = "x2", data = dataset,
                 hue = "y")
plt.show()

# Train test split
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split (x,
        y, test_size = 0.2, random_state = 42)

# Classification analysis using Bagging Meta-Estimator
# technique
from sklearn.ensamble import BaggingClassifier

# We need an estimator for BaggingClassifier
```

```
from sklearn.svm import SVC

# Check for the score of our estimator
svc = SVC()
svc.fit(x_train, y_train)
svc.score(x_train, y_train), svc.score(x_test, y_test)

# Create the BaggingClassifier model
bc = BaggingClassifier(estimator=SVC(), n_estimators=50)
bc.fit(x_train, y_train)

bc.score(x_train, y_train), bc.score(x_test, y_test)

# Classification analysis using Random Forest technique
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n_estimators=50)
rfc.fit(x_train, y_train)

rfc.score(x_train, y_train), rfc.score(x_test, y_test)
```

Regression Analysis:

```
dataset = pd.read_csv("Maths.csv")
dataset.head(3)

# Check for null values
dataset.isnull().sum()

# Remove all the null values
dataset.dropna(inplace=True)
```

```
# Select dependent and independent data
x = dataset [["Number 3"]]
y = dataset ["Total Sum"]

# Train test split
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split (x,
        y, test_size = 0.2 , random_state = 42)

# Regression analysis using Bagging Meta-Estimator
# technique
from sklearn.ensemble import BaggingRegressor

# We need an estimator for BaggingRegressor
from sklearn.linear_model import LinearRegression

# Check for the score of our estimator.
lr = LinearRegression ()
lr.fit (x_train, y_train)
lr.score (x_train, y_train) , lr.score (x_test, y_test)

# Create the BaggingRegressor model
br = BaggingRegressor (estimator = LinearRegression () ,
                       n_estimators = 50)
br.fit (x_train , y_train)

br.score (x_train , y_train) , br.score (x_test, y_test)

# Regression analysis using Random Forest technique
from sklearn.ensemble import RandomForestRegressor
```

```
rfr = RandomForestRegressor(n_estimators=50)
rfr.fit(x_train, y_train)

rfr.score(x_train, y_train), rfr.score(x_test, y_test)
```