Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Автоматизированные системы обработки информации и управления»

Отчет по лабораторной работе № 6

«Ансамбли моделей машинного обучения»

по курсу
“Методы машинного обучения”

Выполнил:
Али Диб А.Ж.
Студент группы ИУ5-22М

Москва, 2020

Цель лабораторной работы

Изучение ансамблей моделей машинного обучения.

```
from datetime import datetime
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import median_absolute_error, r2_score
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
# Enable inline plots
%matplotlib inline
# Set plots formats to save high resolution PNG
from IPython.display import set_matplotlib_formats
set_matplotlib_formats("retina")
%matplotlib inline
```

Набор данных

Наш набор данных касается относительных данных о производительности процессора, опи
цикла, объема памяти и т. Д.

Информация об атрибутах:

1. vendor name
2. Model Name: many unique symbols
3. MYCT: machine cycle time in nanoseconds (integer)
4. MMIN: minimum main memory in kilobytes (integer)
5. MMAX: maximum main memory in kilobytes (integer)
6. CACH: cache memory in kilobytes (integer)
7. CHMIN: minimum channels in units (integer)
8. CHMAX: maximum channels in units (integer)
9. PRP: published relative performance (integer)
10. ERP: estimated relative performance from the original article (integer)

```
:v('https://archive.ics.uci.edu/ml/machine-learning-databases/cpu-performance/machir
```

	vendor_name	Model_Name	MYCT	MMIN	MMAX	CACH	CHMIN	CHMAX	PRP	ERP
0	adviser	32/60	125	256	6000	256	16	128	198	199
1	amdahl	470v/7	29	8000	32000	32	8	32	269	253
2	amdahl	470v/7a	29	8000	32000	32	8	32	220	253
3	amdahl	470v/7b	29	8000	32000	32	8	32	172	253
4	amdahl	470v/7c	29	8000	16000	32	8	16	132	132

```
data.columns
```

```
Index(['vendor_name', 'Model_Name', 'MYCT', 'MMIN', 'MMAX', 'CACH', 'CHMIN',
       'CHMAX', 'PRP', 'ERP'],
      dtype='object')
```

```
data.dtypes
```

```
vendor_name    object
Model_Name     object
MYCT            int64
MMIN            int64
MMAX            int64
CACH            int64
CHMIN           int64
CHMAX           int64
PRP             int64
ERP             int64
dtype: object
```

```
data.isnull().sum()
```

```
vendor_name    0
Model_Name     0
MYCT           0
MMIN           0
MMAX           0
CACH           0
CHMIN          0
CHMAX          0
PRP            0
ERP            0
dtype: int64
```

наш набор данных не содержит пропущенных значений. поэтому не надо обрабатывать их. но мы должны иметь дело со столбцами объектов в нашем наборе данных.

Кодирование категориальных признаков

```
LE=LabelEncoder()
```

```
data_vendor_name=LE.fit_transform(data['vendor_name'])
data['vendor_name']=data_vendor_name


data_Model_Name=LE.fit_transform(data['Model_Name'])
data['Model_Name']=data_Model_Name


data.dtypes
```

```
vendor_name    int64
Model_Name     int64
MYCT           int64
MMIN           int64
MMAX           int64
CACH           int64
CHMIN          int64
CHMAX          int64
PRP            int64
ERP            int64
dtype: object
```

разделиние выборку на обучающую и тестовую.

```
X_train, X_test, y_train, y_test = train_test_split(data[['vendor_name','Model_Name
```

```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(167, 9)
(42, 9)
(167,)
(42,)
```

Выбор метрик для последующей оценки качества модел

мы будем использовать среднюю абсолютную ошибку и медиану абсолютной ошибки

```
def test_model(model):
  return {'mean_absolute_error':mean_absolute_error(y_test, model.predict(X_test)),
          'median_absolute_error' : median_absolute_error(y_test, model.predict(X_t
```

Random Forest Regressor

```
ran_100 = RandomForestRegressor(n_estimators=100)
ran_100.fit(X_train, y_train)
```

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto', max_leaf_nodes=None
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

```
metrics_RF=test_model(ran_100)
metrics_RF
```

```
{'mean_absolute_error': 10.954999999999998, 'median_absolute_error': 2.5}
```

Gradient Boosting Regressor

```
gr_10 = GradientBoostingRegressor(n_estimators=100)
gr_10.fit(X_train, y_train)
```

```
GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                          init=None, learning_rate=0.1, loss='ls', max_depth=3
                          max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=2,
                          min_weight_fraction_leaf=0.0, n_estimators=100,
                          n_iter_no_change=None, presort='deprecated',
                          random_state=None, subsample=1.0, tol=0.0001,
                          validation_fraction=0.1, verbose=0, warm_start=False
```

```
metrics_GBR=test_model(gr_10)
metrics_GBR
```

```
{'mean_absolute_error': 10.228130540371126,
 'median_absolute_error': 1.7911168435883713}
```

Подбор гиперпараметров для выбранных моделей

```
param_grid = {
    'max_depth' : [1, 2, 3, 4, 5],
    'max_samples' : [0.05, 0.1, 0.2, 0.5],
    'max_leaf_nodes':[10, 15],
    'n_estimators':np.array(range(1,100,10))
}
param_grid
```

```
     {'max_depth': [1, 2, 3, 4, 5],
      'max_leaf_nodes': [10, 15],
      'max_samples': [0.05, 0.1, 0.2, 0.5],
```

```
gs = GridSearchCV(RandomForestRegressor(), param_grid,
                  cv=ShuffleSplit(n_splits=10), scoring="neg_mean_squared_error",
                    return_train_score=True, n_jobs=-1)
gs.fit(X_train, y_train)
gs.best_estimator_
```

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=4, max_features='auto', max_leaf_nodes=10,
                      max_samples=0.5, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=71, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

```
gs.best_params_
```

```
{'max_depth': 4, 'max_leaf_nodes': 10, 'max_samples': 0.5, 'n_estimators': 71}
```

```
reg = gs.best_estimator_
reg.fit(X_train, y_train)
new_metrics_RF=test_model(reg)
new_metrics_RF
```

```
{'mean_absolute_error': 15.760364301003799,
 'median_absolute_error': 7.665092253800962}
```

```
param_grid = {
    'max_depth' : [1, 2, 3, 4, 5],
    'max_leaf_nodes':[10, 15],
    'n_estimators':np.array(range(1,100,10))
}
```

```
gbr = GridSearchCV(GradientBoostingRegressor(), param_grid,
                           cv=ShuffleSplit(n_splits=10), scoring="neg_mean_squared
                             return_train_score=True, n_jobs=-1)
gbr.fit(X_train, y_train)
gbr.best_estimator_
```

```
GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                          init=None, learning_rate=0.1, loss='ls', max_depth=4
                          max_features=None, max_leaf_nodes=10,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=2,
                          min_weight_fraction_leaf=0.0, n_estimators=61,
                          n_iter_no_change=None, presort='deprecated',
                          random_state=None, subsample=1.0, tol=0.0001,
                          validation_fraction=0.1, verbose=0, warm_start=False
```

```
reg = gbr.best_estimator_
reg.fit(X_train, y_train)
new_metrics_GBR=test_model(reg)
```

```
new_metrics_GBR=test_model(reg)
new_metrics_GBR
```

```
{'mean_absolute_error': 10.908652611618384,
 'median_absolute_error': 2.9295372090949954}
```

Сравниние качество полученных моделей

```
print(metrics_RF)
print(new_metrics_RF)
```

```
{'mean_absolute_error': 10.954999999999998, 'median_absolute_error': 2.5}
{'mean_absolute_error': 15.760364301003799, 'median_absolute_error': 7.6650922
```

```
print(metrics_GBR)
print(new_metrics_GBR)
```

```
{'mean_absolute_error': 10.228130540371126, 'median_absolute_error': 1.7911168
{'mean_absolute_error': 10.908652611618384, 'median_absolute_error': 2.9295372
```