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»

Выполнила:
Шаххуд Ф.М.
Студентка группы ИУ5И-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
# 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 о лесных пожарах. У нас есть столбцы, которые описывают пожар, и мы
предсказывает площадь пожара с помощью регрессии.

```
data=pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/forest-
data.head()
```

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.0
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.0
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0

```
data.columns
```

```
Index(['X', 'Y', 'month', 'day', 'FFMC', 'DMC', 'DC', 'ISI', 'temp', 'RH',
       'wind', 'rain', 'area'],
      dtype='object')
```

```
data.dtypes
```

```
X          int64
Y          int64
month     object
day       object
FFMC     float64
DMC      float64
DC       float64
ISI      float64
temp     float64
RH         int64
wind     float64
rain     float64
area     float64
dtype: object
```

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

```
X        0
Y        0
month    0
day      0
FFMC     0
DMC      0
DC       0
ISI      0
temp     0
RH       0
wind     0
rain     0
area     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

```
np.unique(data.month)
```

```
array(['apr', 'aug', 'dec', 'feb', 'jan', 'jul', 'jun', 'mar', 'may',
       'nov', 'oct', 'sep'], dtype=object)
```

```
title_mapping = {"jan": 1, "feb": 2, "mar": 3,
                 "apr": 4, "may": 5, "jun": 6, "jul": 7, "aug": 8, "sep": 9,"oct":
                 "nov": 11, "dec": 12 }
data['month'] = data['month'].map(title_mapping)
```

```
np.unique(data.day)
```

```
array(['fri', 'mon', 'sat', 'sun', 'thu', 'tue', 'wed'], dtype=object)
```

```
title_mapping = {"mon": 1, "tue": 2, "wed": 3,
                 "thu": 4, "fri": 5, "sat": 6, "sun": 7 }
data['day'] = data['day'].map(title_mapping)
```

```
data.dtypes
```

```
X           int64
Y           int64
month       int64
day         int64
FFMC      float64
DMC       float64
DC        float64
ISI       float64
temp      float64
RH          int64
wind      float64
rain      float64
area      float64
dtype: object
```

разделиние выборку на обучающую и тестовую.

```
X_train, X_test, y_train, y_test = train_test_split(data[['X','Y','month','day','FF
                        'DC','ISI','temp','RH','wind','rain']],data['area'],test_si
```

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

```
(413, 12)
(104, 12)
(413,)
(104,)
```

Выбор метрик для последующей оценки качества модел

мы будем использовать среднюю абсолютную ошибку и среднюю абсолютную ошибку

```
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': 22.512031153846156,
 'median_absolute_error': 8.625950000000001}
```

Gradient Boosting Regressor

```
gr_10 = GradientBoostingRegressor(n_estimators=10)
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=10,
                          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': 20.983530189424382,
 'median_absolute_error': 8.127994797436074}
```

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],
      'n_estimators': array([ 1, 11, 21, 31, 41, 51, 61, 71, 81, 91])}
```

```
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=1, max_features='auto', max_leaf_nodes=10,
                      max_samples=0.2, 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=31, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

```
gs.best_params_
```

```
{'max_depth': 1, 'max_leaf_nodes': 10, 'max_samples': 0.2, 'n_estimators': 31}
```

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

```
{'mean_absolute_error': 20.133763760414713,
 'median_absolute_error': 7.689291702435838}
```

```
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=3
                          max_features=None, max_leaf_nodes=15,
                          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_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': 22.664977187130713,
 'median_absolute_error': 6.494514329418848}
```

Сравниние качество полученных моделей

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

```
{'mean_absolute_error': 22.512031153846156, 'median_absolute_error': 8.6259500
{'mean_absolute_error': 20.133763760414713, 'median_absolute_error': 7.6892917
```

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

```
{'mean_absolute_error': 20.983530189424382, 'median_absolute_error': 8.1279947
{'mean_absolute_error': 22.664977187130713, 'median_absolute_error': 6.4945143
```

```

```