Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

Отчет по лабораторной работе №4

«Ансамблевые методы»

по дисциплине «Машинное обучение»

Выполнил
студент гр. 3530904/90102 Афанасьев Е.Д.

Руководитель
старший преподаватель ВШПИ Селин И. А.

Санкт-Петербург 2022

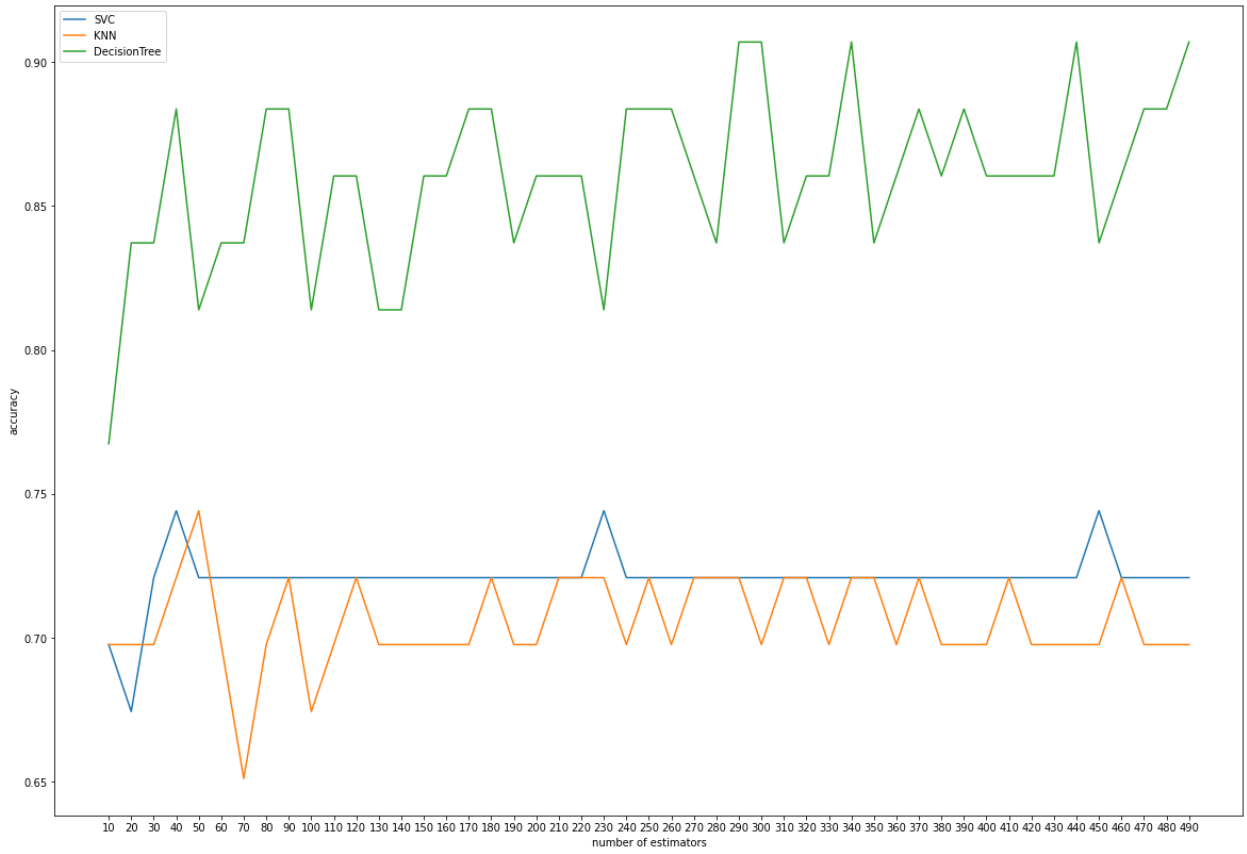
Оглавление

Задачи

1. Исследуйте зависимость качества классификации от количества классификаторов в ансамбле для алгоритмов бэггинга на наборе данных glass.csv с различными базовыми классификаторами. Постройте графики зависимости качества классификации при различном числе классификаторов, объясните полученные результаты.
2. Исследуйте зависимость качества классификации от количества классификаторов в ансамбле для алгоритма бустинга (например, AdaBoost) на наборе данных vehicle.csv с различными базовыми классификаторами. Постройте графики зависимости качества классификации при различном числе классификаторов, объясните полученные результаты.
3. Постройте мета-классификатор для набора данных titanic_train.csv используя стекинг и оцените качество классификации на titanic_train.csv

Исследование зависимости качества классификации от числа классификаторов в бэггинге

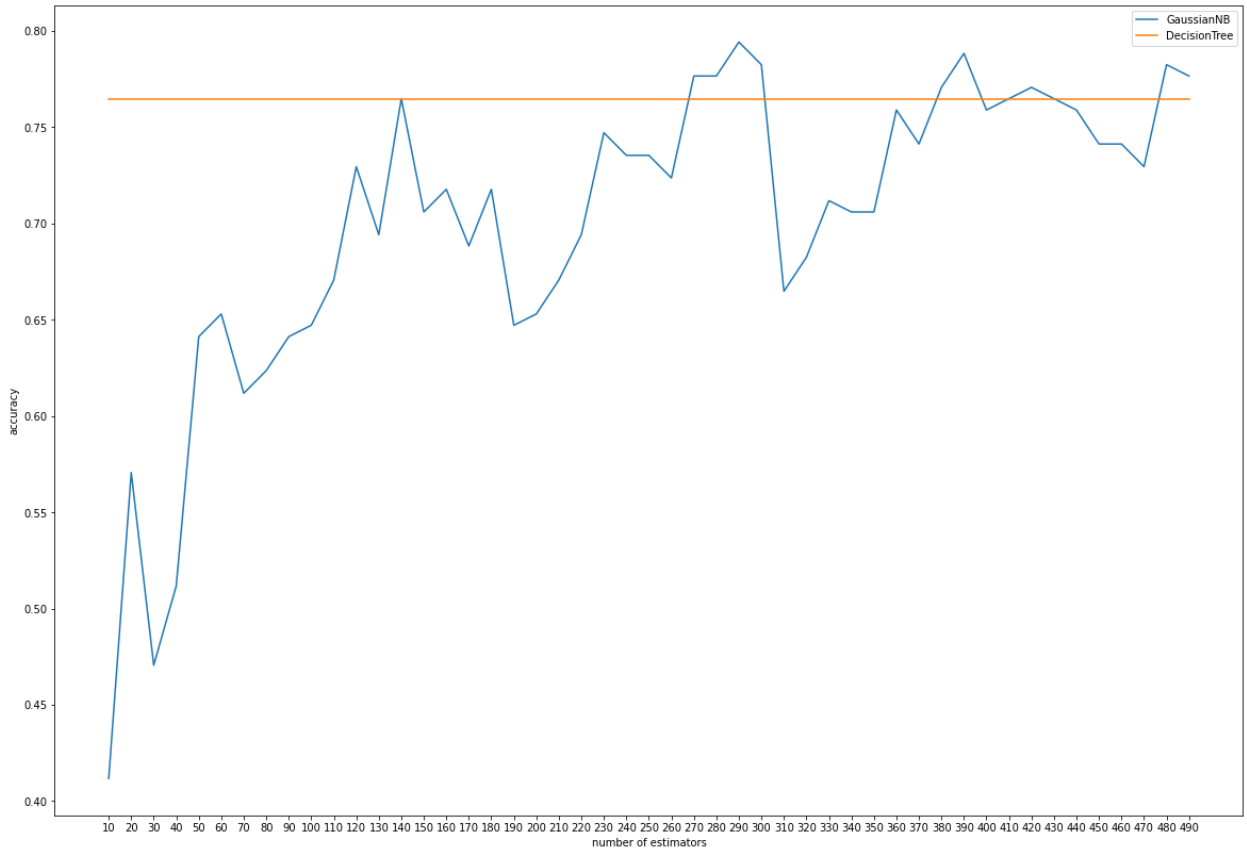
Построим разные модели с различными базовыми классификаторами с числом классификаторов от 10 до 500



Для различных базовых классификаторов были получены графики выше. Можно заметить, что с ростом параметра n_neighbors (количества оценок), точность растет, и достигает какого-то предела, но иногда происходят выбросы, когда нам везет и при меньшем количестве оценок получается большая точность.

Исследование зависимости качества классификации от числа классификаторов в бустинге

Построим разные модели AdaBoostClassifier с различными базовыми классификаторами с числом классификаторов от 10 до 500.



Бустинг с увеличением числа оценок не дал результатов для DecisionTreeClassifier, оценка точности всегда оставалась одинаковой. Для байесовского классификаторов бустинг дал неплохой результат, при увеличении числа соседей растет точность.

Титаник с стекингом

В первую очередь необходимо предобработать данные. Я удалил ненужные столбцы id, имя, билет и кабину, затем перевел категориальные признаки sex и embarked. Затем привел столбец возраста к численному формату, поскольку у него было тип object, с которым не удастся дальше работать. Также заменил пустые значение в столбце Age и Fare на среднее значение во всем столбце и в столбце embarked на самое часто встречающееся значение.

Мне стало интересно проверить точность модели и на тестовой выборке, поэтому я скачал данные с Kaggle и проверил их.

Для стеккинга использовал случайный лес, AdaBoostClassifier и BaggingClassifier с базовыми классификаторами DeciosionTree, KNN и SVC с мета-классификатором LogisticRegression.

```
Accuracy train:  0.9809203142536476
Accuracy test:  0.8014354066985646
```

Была получена модель, которая имеет точность 0.8 на тестовой выборке, что довольно неплохо.

Приложения

1_bagging.py

```
#!/usr/bin/env python
# coding: utf-8


# In[59]:



import numpy as np
import pandas as pd
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')



# In[60]:



data = pd.read_csv('glass.csv')
data = data.drop('Id', axis=1)
X, y = data.drop('Type', axis=1), data.Type
train_X, test_X, train_y, test_y = train_test_split(X, y, train_size=0.8,
random_state=42)
```

```
# In[61]:


estimators = [
    ('SVC', SVC(C=5, gamma='auto')),
    ('KNN', KNeighborsClassifier()),
    ('DecisionTree', DecisionTreeClassifier())
]

n_estimators = np.arange(10, 500, 10)


# In[62]:


accuracy = {
    'SVC': [],
    'KNN': [],
    'DecisionTree': []
}

for title, est in estimators:
    for n in n_estimators:
        clf = BaggingClassifier(base_estimator=est, n_estimators=n).fit(train_X,
train_y)
        pred_y = clf.predict(test_X)


        acc = accuracy_score(test_y, pred_y)
```

```
        accuracy[title].append(acc)


# In[63]:


plt.figure(figsize=(20, 14))
for title, est in estimators:
    plt.plot(accuracy[title], label=title)

plt.xticks(range(len(n_estimators)), n_estimators)
plt.xlabel('number of estimators')
plt.ylabel('accuracy')
plt.legend(loc='best')
```

2_boosting.py

```
#!/usr/bin/env python
# coding: utf-8

# In[14]:


import numpy as np
import pandas as pd
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')


# In[10]:


data = pd.read_csv('vehicle.csv')

X, y = data.drop('Class', axis=1), data.Class
train_X, test_X, train_y, test_y = train_test_split(X, y, train_size=0.8,
random_state=42)


# In[35]:


estimators = [
    ('GaussianNB', GaussianNB()),
    ('DecisionTree', DecisionTreeClassifier())
]

n_estimators = np.arange(10, 500, 10)
```

```
# In[36]:



accuracy = {
    'GaussianNB': [],
    'DecisionTree': []
}

for title, est in estimators:
    for n in n_estimators:
        clf = AdaBoostClassifier(base_estimator=est, n_estimators=n,
random_state=0).fit(train_X, train_y)
        pred_y = clf.predict(test_X)


        acc = accuracy_score(test_y, pred_y)


        accuracy[title].append(acc)



# In[37]:



plt.figure(figsize=(20, 14))
for title, est in estimators:
    plt.plot(accuracy[title], label=title)

plt.xticks(range(len(n_estimators)), n_estimators)
plt.xlabel('number of estimators')
plt.ylabel('accuracy')
```

```
plt.legend(loc='best')
```

3_stacking.py

```
#!/usr/bin/env python
# coding: utf-8

# In[2]:



import numpy as np
import pandas as pd
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier, StackingClassifier,
RandomForestClassifier, BaggingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')



# In[3]:



data_train = pd.read_csv('train.csv')
data_test = pd.read_csv('test.csv')
```

```
data_train = data_train.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'], axis=1)
data_test = data_test.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'], axis=1)

embarked_map = {'S': 1, 'C': 2, 'Q': 3}
sex_map = {'male': -1, 'female': 1}

data_train['Embarked'] = data_train['Embarked'].map(embarked_map)
data_test['Embarked'] = data_test['Embarked'].map(embarked_map)

data_train['Sex'] = data_train['Sex'].map(sex_map)
data_test['Sex'] = data_test['Sex'].map(sex_map)

data_train['Age'] = pd.to_numeric(data_train['Age'], errors='coerce')
data_test['Age'] = pd.to_numeric(data_test['Age'], errors='coerce')

data_train['Age'] = data_train['Age'].fillna(data_train['Age'].mean())
data_test['Age'] = data_test['Age'].fillna(data_test['Age'].mean())

data_train['Embarked'] = data_train['Embarked'].fillna(1)
data_test['Fare'] = data_test['Fare'].fillna(data_test['Fare'].mean())


# In[4]:


X_train, y_train = data_train.drop('Survived', axis=1), data_train.Survived
X_test, y_test = data_test, pd.read_csv('gender_submission.csv').Survived
```

```
# In[26]:



estimators = [
    ('RandomForest', RandomForestClassifier()),
    ('AdaBoost', AdaBoostClassifier()),
    ('BaggingDecisionTree',
BaggingClassifier(base_estimator=DecisionTreeClassifier(), n_estimators=100)),
    ('BaggingSVC', BaggingClassifier(base_estimator=SVC(C=5),
n_estimators=100)),
    ('BaggingKNN', BaggingClassifier(base_estimator=KNeighborsClassifier(),
n_estimators=100)),
]



# In[27]:



clf = StackingClassifier(estimators=estimators,
final_estimator=LogisticRegression())


clf.fit(X_train, y_train)



# In[28]:



y_pred_train = clf.predict(X_train)
y_pred_test = clf.predict(X_test)
```

```
train_acc = accuracy_score(y_train, y_pred_train)
test_acc = accuracy_score(y_test, y_pred_test)

print("Accuracy train: ", train_acc)
print("Accuracy test: ", test_acc)
```