

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



**Отчет**  
**Лабораторная работа № 6**  
**По курсу «Технологии машинного обучения»**  
**«Ансамбли моделей машинного обучения»**

**ИСПОЛНИТЕЛЬ:**

Сергеев И.В.  
Группа ИУ5-64Б

"\_\_" \_\_\_\_\_ 2020 г.

**ПРЕПОДАВАТЕЛЬ:**

Гапанюк Ю.Е.

"\_\_" \_\_\_\_\_ 2020 г.

Москва 2020

## Описание задания

**Цель лабораторной работы** - изучение ансамблей моделей машинного обучения

### Задание:

- Выберите набор данных (датасет) для решения задачи классификации или регрессии.
- В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.
- С использованием метода `train_test_split` разделите выборку на обучающую и тестовую.
- Обучите две ансамблевые модели. Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.

## Текст программы

Программа разрабатывалась в IDE PyCharm. Ниже приведён полный листинг программы:

```
###  
  
import pandas as pd  
from sklearn.model_selection import train_test_split  
from sklearn.model_selection import GridSearchCV  
from sklearn.metrics import accuracy_score  
from sklearn.metrics import classification_report  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.ensemble import GradientBoostingClassifier  
import seaborn as sns  
%matplotlib inline  
sns.set(style="ticks")  
  
###  
  
data = pd.read_csv('dataset.csv', sep=",")  
data.columns  
  
###  
  
data.shape  
  
###  
  
data[' shares'].mean()  
  
###  
  
data.head()
```

```

#%%

df = data[:35000]
df.shape

#%%

answers = []
for row in range(df.shape[0]):
    if df['shares'][row]<100:
        answers.append(0)
    if 100 <= df['shares'][row] < 1000:
        answers.append(1)
    if 1000 <= df['shares'][row] < 10000:
        answers.append(2)
    if df['shares'][row]>=10000:
        answers.append(3)
y = pd.DataFrame(data=answers)
y.head()

#%%

X = df[['n_non_stop_words', 'n_non_stop_unique_tokens',
        'num_hrefs', 'num_self_hrefs', 'num_imgs', 'num_videos',
        'average_token_length', 'num_keywords', 'data_channel_is_lifestyle',
        'data_channel_is_entertainment', 'data_channel_is_bus',
        'data_channel_is_socmed', 'data_channel_is_tech',
        'data_channel_is_world', 'kw_min_min', 'kw_max_min', 'kw_avg_min',
        'kw_min_max', 'kw_max_max', 'kw_avg_max', 'kw_min_avg',
        'kw_max_avg', 'kw_avg_avg', 'self_reference_min_shares',
        'self_reference_max_shares', 'self_reference_avg_shares',
        'weekday_is_monday', 'weekday_is_tuesday', 'weekday_is_wednesday',
        'weekday_is_thursday', 'weekday_is_friday', 'weekday_is_saturday',
        'weekday_is_sunday', 'is_weekend', 'LDA_00', 'LDA_01', 'LDA_02',
        'LDA_03', 'LDA_04', 'global_subjectivity',
        'global_sentiment_polarity', 'global_rate_positive_words',
        'global_rate_negative_words', 'rate_positive_words',
        'rate_negative_words', 'avg_positive_polarity',
        'min_positive_polarity', 'max_positive_polarity',
        'avg_negative_polarity', 'min_negative_polarity',
        'max_negative_polarity', 'title_subjectivity',
        'title_sentiment_polarity', 'abs_title_subjectivity',
        'abs_title_sentiment_polarity']]
X.head()

#%%

df.isnull().sum()

#%%

X_train,X_test,y_train,y_test = train_test_split(X,y, test_size=0.3, random_state =
1)

#%%

forest_model = RandomForestClassifier()
forest_model.fit(X_train,y_train)
forest_prediction = forest_model.predict(X_test)
print("Accuracy :\t",accuracy_score(y_test, forest_prediction))
print("Classification report:\n",classification_report(y_test, forest_prediction))

```

```

#%%

parameters = {
    'n_estimators'      : [10,20,50,100,200,500],
    'max_depth'         : [8, 9, 10, 11, 12],
    'random_state'      : [0],
}

clf = GridSearchCV(RandomForestClassifier(), parameters, cv=10, n_jobs=-1)
clf.fit(X_train, y_train)

print(clf.score(X_train, y_train))
print(clf.best_params_)

#%%

forest_best_model =
RandomForestClassifier(n_estimators=50,max_depth=9,random_state=0)
forest_best_model.fit(X_train,y_train)
forest_best_prediction = forest_best_model.predict(X_test)
print("Accuracy :\t",accuracy_score(y_test, forest_best_prediction))
print("Классификации отчет:\n",classification_report(y_test, forest_best_prediction))

#%%

gb_model = GradientBoostingClassifier()
gb_model.fit(X_train,y_train)
gb_prediction = gb_model.predict(X_test)
print("Accuracy :\t",accuracy_score(y_test, gb_prediction))
print("Classification report:\n",classification_report(y_test, gb_prediction))

#%%

parameters = {
    "learning_rate"      : [0.01, 0.025, 0.05, 0.075, 0.1, 0.15, 0.2],
    "subsample": [0.5, 0.618, 0.8, 0.85, 0.9, 0.95, 1.0],
    'n_estimators'      : [10],
    'max_depth'         : [3],
    'random_state'      : [0],
}

clf_gb = GridSearchCV(GradientBoostingClassifier(), parameters, cv=10, n_jobs=-1)
clf_gb.fit(X_train, y_train)

print(clf_gb.score(X_train, y_train))
print(clf_gb.best_params_)

#%%

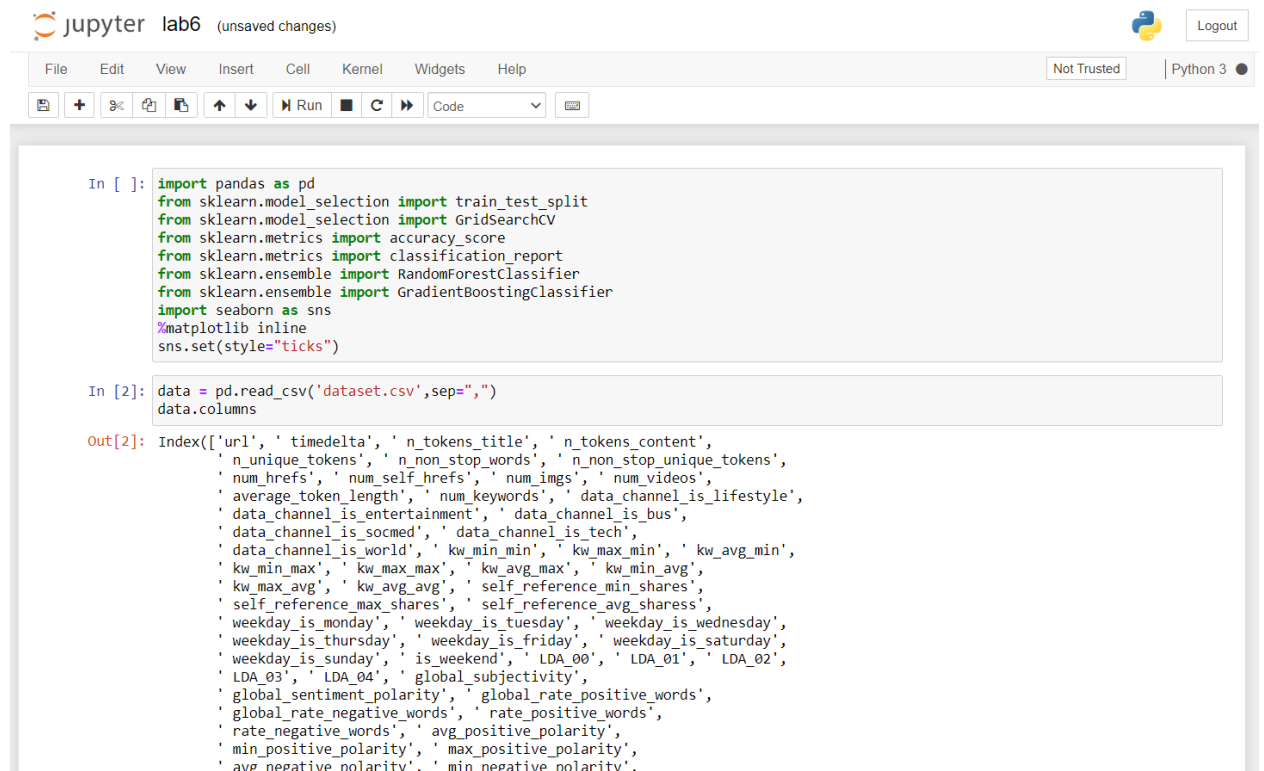
gb_best_model =
GradientBoostingClassifier(learning_rate=0.15,subsample=0.618,n_estimators=10,max_dep
th=3,random_state=0)
gb_best_model.fit(X_train,y_train)
gb_best_prediction = gb_best_model.predict(X_test)
print("Accuracy :\t",accuracy_score(y_test, gb_best_prediction))
print("Classification report:\n",classification_report(y_test, gb_best_prediction))

#%%

```

## Примеры выполнения программы

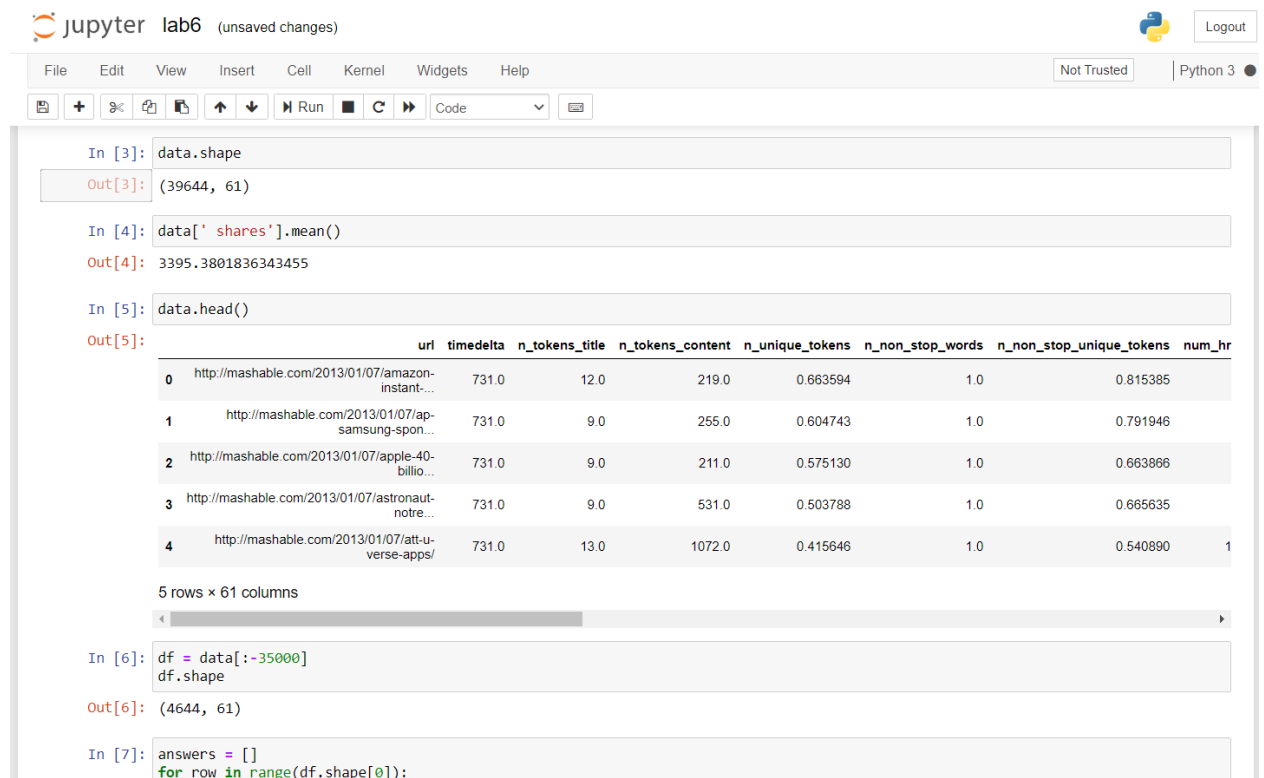
Выполнение программы, а также наглядная демонстрация входных и выходных данных (таблиц, графиков и тд) осуществлялась на базе Jupyter Notebook, сервер которого запускался из-под PyCharm. Ниже приведены скриншоты, отражающие работу программы:



```
In [ ]: import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
import seaborn as sns
%matplotlib inline
sns.set(style="ticks")

In [2]: data = pd.read_csv('dataset.csv', sep=",")
data.columns

Out[2]: Index(['url', 'timedelta', 'n_tokens_title', 'n_tokens_content',
              'n_unique_tokens', 'n_non_stop_words', 'n_non_stop_unique_tokens',
              'num_hrefs', 'num_self_hrefs', 'num_imgs', 'num_videos',
              'average_token_length', 'num_keywords', 'data_channel_is_lifestyle',
              'data_channel_is_entertainment', 'data_channel_is_bus',
              'data_channel_is_socmed', 'data_channel_is_tech',
              'data_channel_is_world', 'kw_min_min', 'kw_max_min', 'kw_avg_min',
              'kw_min_max', 'kw_max_max', 'kw_avg_max', 'kw_min_avg',
              'kw_max_avg', 'kw_avg_avg', 'self_reference_min_shares',
              'self_reference_max_shares', 'self_reference_avg_shares',
              'weekday_is_monday', 'weekday_is_tuesday', 'weekday_is_wednesday',
              'weekday_is_thursday', 'weekday_is_friday', 'weekday_is_saturday',
              'weekday_is_sunday', 'is_weekend', 'LDA_00', 'LDA_01', 'LDA_02',
              'LDA_03', 'LDA_04', 'global_subjectivity',
              'global_sentiment_polarity', 'global_rate_positive_words',
              'global_rate_negative_words', 'rate_positive_words',
              'rate_negative_words', 'avg_positive_polarity',
              'min_positive_polarity', 'max_positive_polarity',
              'avg_negative_polarity', 'min_negative_polarity',
```



```
In [3]: data.shape

Out[3]: (39644, 61)

In [4]: data['shares'].mean()

Out[4]: 3395.3801836343455

In [5]: data.head()

Out[5]:
```


	uri	timedelta	n_tokens_title	n_tokens_content	n_unique_tokens	n_non_stop_words	n_non_stop_unique_tokens	num_hr
0	http://mashable.com/2013/01/07/amazon-instant...	731.0	12.0	219.0	0.663594	1.0	0.815385	
1	http://mashable.com/2013/01/07/ap-samsung-spon...	731.0	9.0	255.0	0.604743	1.0	0.791946	
2	http://mashable.com/2013/01/07/apple-40-billio...	731.0	9.0	211.0	0.575130	1.0	0.663866	
3	http://mashable.com/2013/01/07/astronaut-notre...	731.0	9.0	531.0	0.503788	1.0	0.665635	
4	http://mashable.com/2013/01/07/att-u-verse-apps/	731.0	13.0	1072.0	0.415646	1.0	0.540890	1


5 rows x 61 columns

```
In [6]: df = data[:35000]
df.shape

Out[6]: (4644, 61)











In [7]: answers = []
for row in range(df.shape[0]):
```


**jupyter lab6** (unsaved changes)
 


 Logout

File Edit View Insert Cell Kernel Widgets Help
 

Not Trusted Python 3

In [7]:

```

answers = []
for row in range(df.shape[0]):
    if df['shares'][row]<100:
        answers.append(0)
    if 100 <= df['shares'][row] < 1000:
        answers.append(1)
    if 1000 <= df['shares'][row] < 10000:
        answers.append(2)
    if df['shares'][row]>=10000:
        answers.append(3)
y = pd.DataFrame(data=answers)
y.head()

```

Out[7]:


0
0 1
1 1
2 2
3 2
4 1


In [8]:

```

x = df[['n_non_stop_words', 'n_non_stop_unique_tokens',
        'num_hrefs', 'num_self_hrefs', 'num_imgs', 'num_videos',
        'average_token_length', 'num_keywords', 'data_channel_is_lifestyle',
        'data_channel_is_entertainment', 'data_channel_is_bus',
        'data_channel_is_socmed', 'data_channel_is_tech',
        'data_channel_is_world', 'kw_min_min', 'kw_max_min', 'kw_avg_min',
        'kw_min_max', 'kw_max_max', 'kw_avg_max', 'kw_min_avg',
        'kw_max_avg', 'kw_avg_avg', 'self_reference_min_shares',
        'self_reference_max_shares', 'self_reference_avg_shares']]











```


**jupyter lab6** (unsaved changes)
 


 Logout

File Edit View Insert Cell Kernel Widgets Help
 

Not Trusted Python 3

In [8]:

```

x = df[['n_non_stop_words', 'n_non_stop_unique_tokens',
        'num_hrefs', 'num_self_hrefs', 'num_imgs', 'num_videos',
        'average_token_length', 'num_keywords', 'data_channel_is_lifestyle',
        'data_channel_is_entertainment', 'data_channel_is_bus',
        'data_channel_is_socmed', 'data_channel_is_tech',
        'data_channel_is_world', 'kw_min_min', 'kw_max_min', 'kw_avg_min',
        'kw_min_max', 'kw_max_max', 'kw_avg_max', 'kw_min_avg',
        'kw_max_avg', 'kw_avg_avg', 'self_reference_min_shares',
        'self_reference_max_shares', 'self_reference_avg_shares',
        'weekday_is_monday', 'weekday_is_tuesday', 'weekday_is_wednesday',
        'weekday_is_thursday', 'weekday_is_friday', 'weekday_is_saturday',
        'weekday_is_sunday', 'is_weekend', 'LDA_00', 'LDA_01', 'LDA_02',
        'LDA_03', 'LDA_04', 'global_subjectivity',
        'global_sentiment_polarity', 'global_rate_positive_words',
        'global_rate_negative_words', 'rate_positive_words',
        'rate_negative_words', 'avg_positive_polarity',
        'min_positive_polarity', 'max_positive_polarity',
        'avg_negative_polarity', 'min_negative_polarity',
        'max_negative_polarity', 'title_subjectivity',
        'title_sentiment_polarity', 'abs_title_subjectivity',
        'abs_title_sentiment_polarity']]
x.head()

```

Out[8]:

	n_non_stop_words	n_non_stop_unique_tokens	num_hrefs	num_self_hrefs	num_imgs	num_videos	average_token_length	num_keywords	data_channel_is_
0	1.0	0.815385	4.0	2.0	1.0	0.0	4.680365	5.0	
1	1.0	0.791946	3.0	1.0	1.0	0.0	4.913725	4.0	
2	1.0	0.663866	3.0	1.0	1.0	0.0	4.393365	6.0	
3	1.0	0.665635	9.0	0.0	1.0	0.0	4.404896	7.0	
4	1.0	0.540890	19.0	19.0	20.0	0.0	4.682836	7.0	

5 rows x 55 columns