# Лабораторная работа №3

## Выгрузка данных из ЛР №2 (вариант №8)

## ('comp.sys.mac.hardware', 'soc.religion.christian', 'talk.religion.misc')

In [2]:
```
import warnings
from sklearn.datasets import fetch_20newsgroups
warnings.simplefilter(action='ignore', category=FutureWarning)
```

In [3]:
```
categories = ['comp.sys.mac.hardware', 'soc.religion.christian', 'talk.religion.misc']
remove = ('headers', 'footers', 'quotes')

twenty_train_full = fetch_20newsgroups(subset='train', categories=categories, shuffle=True, random_state=42, re
twenty_test_full = fetch_20newsgroups(subset='test', categories=categories, shuffle=True, random_state=42, remo
```

## Применение стемминга

In [27]:
```
import nltk
from nltk import word_tokenize
from nltk.stem import *

nltk.download('punkt')
```

In [5]:
```
def stemming(data):
    porter_stemmer = PorterStemmer()
    stem = []
    for text in data:
        nltk_tokens = word_tokenize(text)
        line = ''.join([' ' + porter_stemmer.stem(word) for word in nltk_tokens])
        stem.append(line)
    return stem
```

In [6]:
```
stem_train = stemming(twenty_train_full.data)
stem_test = stemming(twenty_test_full.data)
```

## Задание

## Вариант №8

## Методы: [SVM, DT, LR]

In [7]:
```
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
```

In [8]:
```
stop_words = [None, 'english']
max_features_values = [100, 500, 1000, 5000, 10000]
use_idf = [True, False]
```

In [9]:
```
dt_first = range(1, 5, 1)
dt_second = range(5, 100, 20)

decision_tree_max_depth = [*dt_first, *dt_second]
```

In [10]:
```
parameters_svm = {
    'vect__max_features': max_features_values,
    'vect__stop_words': stop_words,
    'tfidf__use_idf': use_idf,
}

parameters_dt = {
    'vect__max_features': max_features_values,
    'vect__stop_words': stop_words,
    'tfidf__use_idf': use_idf,
    'clf__criterion': ('gini', 'entropy'),
    'clf__max_depth': decision_tree_max_depth,
}

parameters_lr = {
    'vect__max_features': max_features_values,
    'vect__stop_words': stop_words,
    'tfidf__use_idf': use_idf,
    'clf__solver': ['newton-cg', 'lbfgs', 'sag', 'liblinear'],
    'clf__penalty': ['l2']
}
```

```
parameters_lr_l1 = {
    'vect__max_features': max_features_values,
    'vect__stop_words': stop_words,
    'tfidf__use_idf': use_idf,
    'clf__solver': ['liblinear'],
    'clf__penalty': ['l1'],
}
```

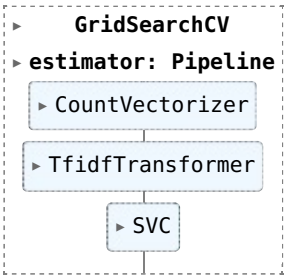
In [11]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
```

## Метод опорных векторов (SVM)

### Без использования стемминга

In [12]:
```
text_clf_svm = Pipeline([('vect', CountVectorizer()),
                         ('tfidf', TfidfTransformer()),
                         ('clf', SVC())])
gscv_svm = GridSearchCV(text_clf_svm, param_grid=parameters_svm, n_jobs=-1)
gscv_svm.fit(twenty_train_full.data, twenty_train_full.target)
```

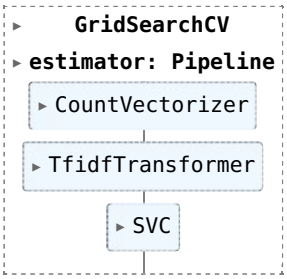
Out[12]:




### С использованием стемминга

In [13]:
```
text_clf_svm_stem = Pipeline([('vect', CountVectorizer()),
                              ('tfidf', TfidfTransformer()),
                              ('clf', SVC(kernel='linear', C=1.0))])
gscv_svm_stem = GridSearchCV(text_clf_svm_stem, param_grid=parameters_svm, n_jobs=-1)
gscv_svm_stem.fit(stem_train, twenty_train_full.target)
```

Out[13]:

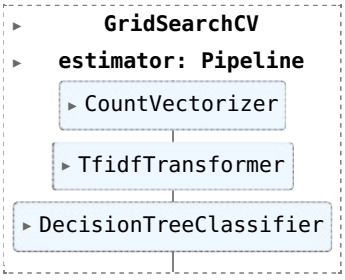



## Дерево решений (DT)

### Без использования стемминга

In [14]:
```
text_clf_dt = Pipeline([('vect', CountVectorizer()),
                        ('tfidf', TfidfTransformer()),
                        ('clf', DecisionTreeClassifier())])
gscv_dt = GridSearchCV(text_clf_dt, param_grid=parameters_dt, n_jobs=-1)
gscv_dt.fit(twenty_train_full.data, twenty_train_full.target)
```

Out[14]:

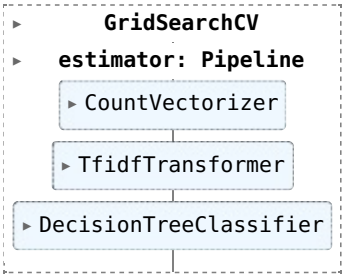



### С использованием стема

In [15]:
```
text_clf_dt_stem = Pipeline([('vect', CountVectorizer()),
                             ('tfidf', TfidfTransformer()),
```

```
                                ('clf', DecisionTreeClassifier())])
gscv_dt_stem = GridSearchCV(text_clf_dt_stem, param_grid=parameters_dt, n_jobs=-1)
gscv_dt_stem.fit(stem_train, twenty_train_full.target)
```

Out[15]:




## Логистическая регрессия (LR)

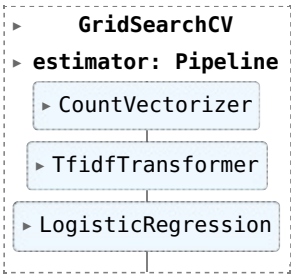
### Без использования стемминга

In [16]:

```
text_clf_lr = Pipeline([('vect', CountVectorizer()),
                        ('tfidf', TfidfTransformer()),
                        ('clf', LogisticRegression())])
gscv_lr = GridSearchCV(text_clf_lr, param_grid=parameters_lr, n_jobs=-1)
gscv_lr.fit(twenty_train_full.data, twenty_train_full.target)


text_clf_lr_l1 = Pipeline([('vect', CountVectorizer()),
                        ('tfidf', TfidfTransformer()),
                        ('clf', LogisticRegression())])
gscv_lr_l1 = GridSearchCV(text_clf_lr_l1, param_grid=parameters_lr_l1, n_jobs=-1)
gscv_lr_l1.fit(twenty_train_full.data, twenty_train_full.target)
```

Out[16]:



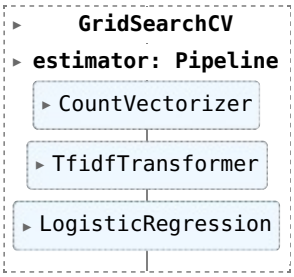

### С использованием стемминга

In [17]:

```
text_clf_lr_stem = Pipeline([('vect', CountVectorizer()),
                             ('tfidf', TfidfTransformer()),
                             ('clf', LogisticRegression())])
gscv_lr_stem = GridSearchCV(text_clf_lr_stem, param_grid=parameters_lr, n_jobs=-1)
gscv_lr_stem.fit(stem_train, twenty_train_full.target)


text_clf_lr_l1_stem = Pipeline([('vect', CountVectorizer()),
                             ('tfidf', TfidfTransformer()),
                             ('clf', LogisticRegression())])
gscv_lr_l1_stem = GridSearchCV(text_clf_lr_l1_stem, param_grid=parameters_lr_l1, n_jobs=-1)
gscv_lr_l1_stem.fit(stem_train, twenty_train_full.target)
```

Out[17]:




## Вывод полученных результатов анализа

In [18]:

```
from sklearn.metrics import classification_report
```

In [19]:

```
predicted_svm = gscv_svm.predict(twenty_test_full.data)
print('Метод опорных векторов (SVM) без стемминга\n')
print(classification_report(twenty_test_full.target, predicted_svm, target_names=categories))
print(gscv_svm.best_params_)
```

```
Метод опорных векторов (SVM) без стемминга

                        precision    recall  f1-score   support

 comp.sys.mac.hardware       0.86      0.98      0.92       385
soc.religion.christian       0.74      0.88      0.80       398
    talk.religion.misc       0.86      0.41      0.56       251

              accuracy                           0.80      1034
             macro avg       0.82      0.76      0.76      1034
          weighted avg       0.81      0.80      0.79      1034

{'tfidf__use_idf': True, 'vect__max_features': 5000, 'vect__stop_words': None}
```

In [20]:
```
predicted_svm_stem = gscv_svm_stem.predict(twenty_test_full.data)
print('Метод опорных векторов (SVM) со стеммингом\n')
print(classification_report(twenty_test_full.target, predicted_svm_stem, target_names=categories))
print(gscv_svm_stem.best_params_)
```

```
Метод опорных векторов (SVM) со стеммингом

                        precision    recall  f1-score   support

 comp.sys.mac.hardware       0.88      0.92      0.90       385
soc.religion.christian       0.77      0.71      0.74       398
    talk.religion.misc       0.57      0.60      0.59       251

              accuracy                           0.76      1034
             macro avg       0.74      0.74      0.74      1034
          weighted avg       0.76      0.76      0.76      1034

{'tfidf__use_idf': True, 'vect__max_features': 10000, 'vect__stop_words': 'english'}
```

In [21]:
```
predicted_dt = gscv_dt.predict(twenty_test_full.data)
print('Дерево решений (DT) без стемминга\n')
print(classification_report(twenty_test_full.target, predicted_dt, target_names=categories))
print(gscv_dt.best_params_)
```

```
Дерево решений (DT) без стемминга

                        precision    recall  f1-score   support

 comp.sys.mac.hardware       0.86      0.71      0.78       385
soc.religion.christian       0.66      0.63      0.64       398
    talk.religion.misc       0.36      0.49      0.42       251

              accuracy                           0.62      1034
             macro avg       0.63      0.61      0.61      1034
          weighted avg       0.66      0.62      0.64      1034

{'clf__criterion': 'gini', 'clf__max_depth': 65, 'tfidf__use_idf': False, 'vect__max_features': 5000, 'vect__st
op_words': 'english'}
```

In [22]:
```
predicted_dt_stem = gscv_dt_stem.predict(twenty_test_full.data)
print('Дерево решений (DT) со стеммингом\n')
print(classification_report(twenty_test_full.target, predicted_dt_stem, target_names=categories))
print(gscv_dt_stem.best_params_)
```

```
Дерево решений (DT) со стеммингом

                        precision    recall  f1-score   support

 comp.sys.mac.hardware       0.84      0.62      0.71       385
soc.religion.christian       0.67      0.63      0.65       398
    talk.religion.misc       0.34      0.51      0.41       251

              accuracy                           0.60      1034
             macro avg       0.62      0.59      0.59      1034
          weighted avg       0.65      0.60      0.61      1034

{'clf__criterion': 'gini', 'clf__max_depth': 45, 'tfidf__use_idf': False, 'vect__max_features': 10000, 'vect__s
top_words': 'english'}
```

In [23]:
```
predicted_lr = gscv_lr.predict(twenty_test_full.data)
print('Логистическая регрессия (LR) без стемминга\n')
print(classification_report(twenty_test_full.target, predicted_lr, target_names=categories))
print(gscv_lr.best_params_)

predicted_lr_l1 = gscv_lr_l1.predict(twenty_test_full.data)
print('Логистическая регрессия_l1 (LR) без стемминга\n')
print(classification_report(twenty_test_full.target, predicted_lr_l1, target_names=categories))
print(gscv_lr_l1.best_params_)
```

```
Логистическая регрессия (LR) без стемминга

                        precision    recall  f1-score   support

 comp.sys.mac.hardware       0.88      0.98      0.93       385
soc.religion.christian       0.73      0.89      0.80       398
    talk.religion.misc       0.80      0.39      0.52       251

              accuracy                           0.80      1034
             macro avg       0.80      0.75      0.75      1034
          weighted avg       0.80      0.80      0.78      1034

{'clf__penalty': 'l2', 'clf__solver': 'newton-cg', 'tfidf__use_idf': True, 'vect__max_features': 10000, 'vect
__stop_words': 'english'}
Логистическая регрессия_l1 (LR) без стемминга

                        precision    recall  f1-score   support

 comp.sys.mac.hardware       0.90      0.88      0.89       385
soc.religion.christian       0.70      0.79      0.74       398
    talk.religion.misc       0.50      0.41      0.45       251

              accuracy                           0.73      1034
             macro avg       0.70      0.69      0.69      1034
          weighted avg       0.73      0.73      0.73      1034

{'clf__penalty': 'l1', 'clf__solver': 'liblinear', 'tfidf__use_idf': True, 'vect__max_features': 500, 'vect__
stop_words': 'english'}
```

In [24]:
```
predicted_lr_stem = gscv_lr_stem.predict(twenty_test_full.data)
print('Логистическая регрессия (LR) со стеммингом\n')
print(classification_report(twenty_test_full.target, predicted_lr_stem, target_names=categories))
print(gscv_lr_stem.best_params_)

predicted_lr_l1_stem = gscv_lr_l1_stem.predict(twenty_test_full.data)
print('Логистическая регрессия_l1 (LR) со стеммингом\n')
print(classification_report(twenty_test_full.target, predicted_lr_l1_stem, target_names=categories))
print(gscv_lr_l1_stem.best_params_)
```

```
Логистическая регрессия (LR) со стеммингом

                        precision    recall  f1-score   support

 comp.sys.mac.hardware       0.81      0.97      0.88       385
soc.religion.christian       0.77      0.75      0.76       398
    talk.religion.misc       0.65      0.48      0.56       251

              accuracy                           0.77      1034
             macro avg       0.75      0.73      0.73      1034
          weighted avg       0.76      0.77      0.76      1034

{'clf__penalty': 'l2', 'clf__solver': 'newton-cg', 'tfidf__use_idf': True, 'vect__max_features': 5000, 'vect__s
top_words': 'english'}
Логистическая регрессия_l1 (LR) со стеммингом

                        precision    recall  f1-score   support

 comp.sys.mac.hardware       0.84      0.81      0.82       385
soc.religion.christian       0.71      0.67      0.69       398
    talk.religion.misc       0.44      0.51      0.48       251

              accuracy                           0.68      1034
             macro avg       0.67      0.66      0.66      1034
          weighted avg       0.70      0.68      0.69      1034

{'clf__penalty': 'l1', 'clf__solver': 'liblinear', 'tfidf__use_idf': True, 'vect__max_features': 500, 'vect__st
op_words': 'english'}
```

## Сравнительная таблица

In [25]:
```
import pandas as pd
```

In [26]:
```
writer = pd.ExcelWriter('result.xlsx', engine='openpyxl')

# Метод опорных векторов (SVM) без стемминга
df1 = pd.DataFrame(classification_report(predicted_svm, twenty_test_full.target, output_dict=True))

# Метод опорных векторов (SVM) со стеммингом
df2 = pd.DataFrame(classification_report(predicted_svm_stem, twenty_test_full.target, output_dict=True))

# Дерево решений (DT) без стемминга
df3 = pd.DataFrame(classification_report(predicted_dt, twenty_test_full.target, output_dict=True))

# Дерево решений (DT) со стеммингом
df4 = pd.DataFrame(classification_report(predicted_dt_stem, twenty_test_full.target, output_dict=True))

# Логистическая регрессия (LR) без стемминга
```

```
df5 = pd.DataFrame(classification_report(predicted_lr, twenty_test_full.target, output_dict=True))

# Логистическая регрессия_l1 (LR) без стемминга
df6 = pd.DataFrame(classification_report(predicted_lr_l1, twenty_test_full.target, output_dict=True))

# Логистическая регрессия (LR) со стеммингом
df7 = pd.DataFrame(classification_report(predicted_lr_stem, twenty_test_full.target, output_dict=True))

# Логистическая регрессия_l1 (LR) со стеммингом
df8 = pd.DataFrame(classification_report(predicted_lr_l1_stem, twenty_test_full.target, output_dict=True))

df1.to_excel(writer, sheet_name='SVM без стемминга')
df2.to_excel(writer, sheet_name='SVM со стеммингом')

df3.to_excel(writer, sheet_name='DT без стемминга')
df4.to_excel(writer, sheet_name='DT со стеммингом')

df5.to_excel(writer, sheet_name='LR без стемминга')
df6.to_excel(writer, sheet_name='LR_l1 без стемминга')

df7.to_excel(writer, sheet_name='LR со стеммингом')
df8.to_excel(writer, sheet_name='LR_l1 со стеммингом')

writer.close()
```