包：

numpy, pandas, matplotlib
sklearn, nltk, gensim, keras

代码：

1. 20_Groups_Data_Analysis.ipynb 是分析“20类新闻包”的代码，结论用于之前的开题报告中；
2. 4_Groups_Data_Analysis.ipynb 是分析项目实践中“4类新闻包”的代码；
3. Machine_Learning_Models.ipynb 是TF-IDF词袋模型特征工程、五种机器学习模型的代码；
4. Deep_Learning_Models.ipynb 是基于text8数据包的词向量模型的特征工程、两种深度学习模型的代码；
5. Validate_on_other_datasets.ipynb 是在另外一个“4类新闻包”数据集上验证朴素贝叶斯模型、卷积神经网络的代码。

其他：

1. text8是用于训练词嵌入模型的数据包；
2. Project Report是项目报告；
3. models文件夹，子文件夹machine_learning夹保存了五个机器学习模型，子文件夹deep_learning保存了词嵌入模型和两个深度学习模型；
4. helper.py和visualizer.py提供一些会重复使用的方法；

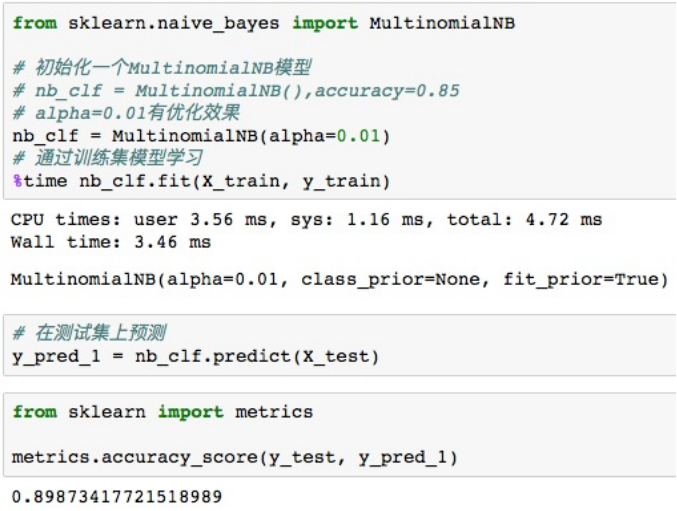
模型训练时长：

机器学习模型	朴素贝叶斯	k近邻	支持向量机	提升树	随机森林
训练时长	3.46ms	1.72ms	48.1ms	18min4s	1.99s

深度学习模型	卷积神经网络	LSTM-循环神经网络
训练时长	3min17s	16min1s

运行成功截图：

朴素贝叶斯模型



```
from sklearn.naive_bayes import MultinomialNB

# 初始化一个MultinomialNB模型
# nb_clf = MultinomialNB(),accuracy=0.85
# alpha=0.01有优化效果
nb_clf = MultinomialNB(alpha=0.01)
# 通过训练集模型学习
%time nb_clf.fit(X_train, y_train)
```

```
CPU times: user 3.56 ms, sys: 1.16 ms, total: 4.72 ms
Wall time: 3.46 ms

MultinomialNB(alpha=0.01, class_prior=None, fit_prior=True)
```

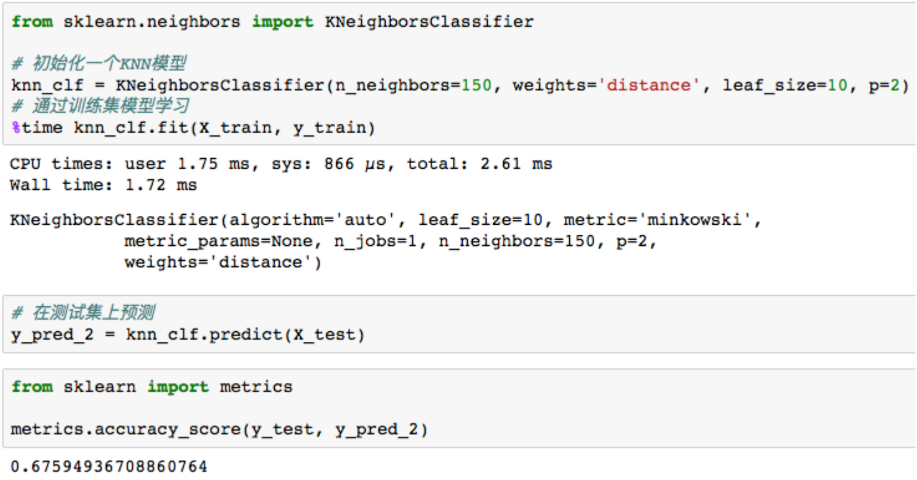
```
# 在测试集上预测
y_pred_1 = nb_clf.predict(X_test)
```

```
from sklearn import metrics

metrics.accuracy_score(y_test, y_pred_1)
```

```
0.89873417721518989
```

k近邻模型



```
from sklearn.neighbors import KNeighborsClassifier

# 初始化一个KNN模型
knn_clf = KNeighborsClassifier(n_neighbors=150, weights='distance', leaf_size=10, p=2)
# 通过训练集模型学习
%time knn_clf.fit(X_train, y_train)
```

```
CPU times: user 1.75 ms, sys: 866 µs, total: 2.61 ms
Wall time: 1.72 ms
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=10, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=150, p=2,
           weights='distance')
```

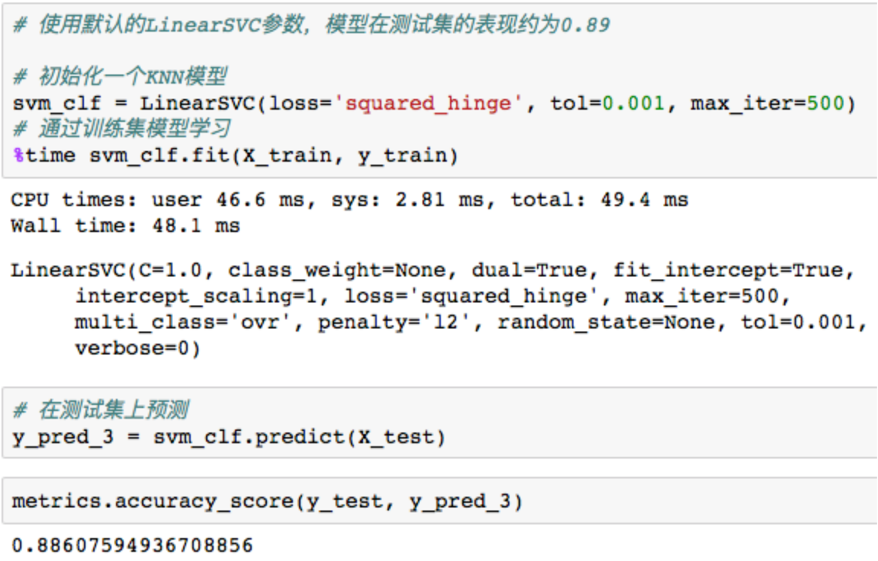
```
# 在测试集上预测
y_pred_2 = knn_clf.predict(X_test)
```

```
from sklearn import metrics

metrics.accuracy_score(y_test, y_pred_2)
```

```
0.67594936708860764
```

支持向量机模型



```
# 使用默认的LinearSVC参数，模型在测试集的表现约为0.89

# 初始化一个KNN模型
svm_clf = LinearSVC(loss='squared_hinge', tol=0.001, max_iter=500)
# 通过训练集模型学习
%time svm_clf.fit(X_train, y_train)
```

```
CPU times: user 46.6 ms, sys: 2.81 ms, total: 49.4 ms
Wall time: 48.1 ms
```

```
LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
     intercept_scaling=1, loss='squared_hinge', max_iter=500,
     multi_class='ovr', penalty='l2', random_state=None, tol=0.001,
     verbose=0)
```

```
# 在测试集上预测
y_pred_3 = svm_clf.predict(X_test)
```

```
metrics.accuracy_score(y_test, y_pred_3)
```

```
0.8860759493670886
```

提升树模型

```
# 优化参数，在Pycharm中运行
# learning_rate=0.12, n_estimators=200, max_depth=5, min_samples_split=2, min_samples_leaf=2
from sklearn.ensemble import GradientBoostingClassifier

gbdt_clf_2 = GradientBoostingClassifier(learning_rate=0.12, n_estimators=200,
                                        max_depth=5, min_samples_split=2, min_samples_leaf=2)
%time gbdt_clf_2.fit(X_train.toarray(), y_train)
```

```
CPU times: user 17min 59s, sys: 2.75 s, total: 18min 2s
Wall time: 18min 4s
```

```
GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.12, loss='deviance', max_depth=5,
              max_features=None, max_leaf_nodes=None,
              min_impurity_split=1e-07, min_samples_leaf=2,
              min_samples_split=2, min_weight_fraction_leaf=0.0,
              n_estimators=200, presort='auto', random_state=None,
              subsample=1.0, verbose=0, warm_start=False)
```

```
gbdt_pred_2 = gbdt_clf_2.predict(X_test.toarray())
```

```
print(metrics.accuracy_score(y_test, gbdt_pred_2))
print(metrics.f1_score(y_test, gbdt_pred_2, average='macro'))
print(metrics.precision_score(y_test, gbdt_pred_2, average='macro'))
print(metrics.recall_score(y_test, gbdt_pred_2, average='macro'))
```

```
0.820886075949
0.821040898643
0.82355489049
0.820902844217
```

随机森林模型

```
from sklearn.ensemble import RandomForestClassifier

rf_clf = RandomForestClassifier(n_estimators=300, min_samples_split=2, min_samples_leaf=3)

%time rf_clf.fit(X_train, y_train)
```

```
CPU times: user 1.96 s, sys: 10.6 ms, total: 1.97 s
Wall time: 1.99 s
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=3,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators=300, n_jobs=1, oob_score=False, random_state=None,
            verbose=0, warm_start=False)
```

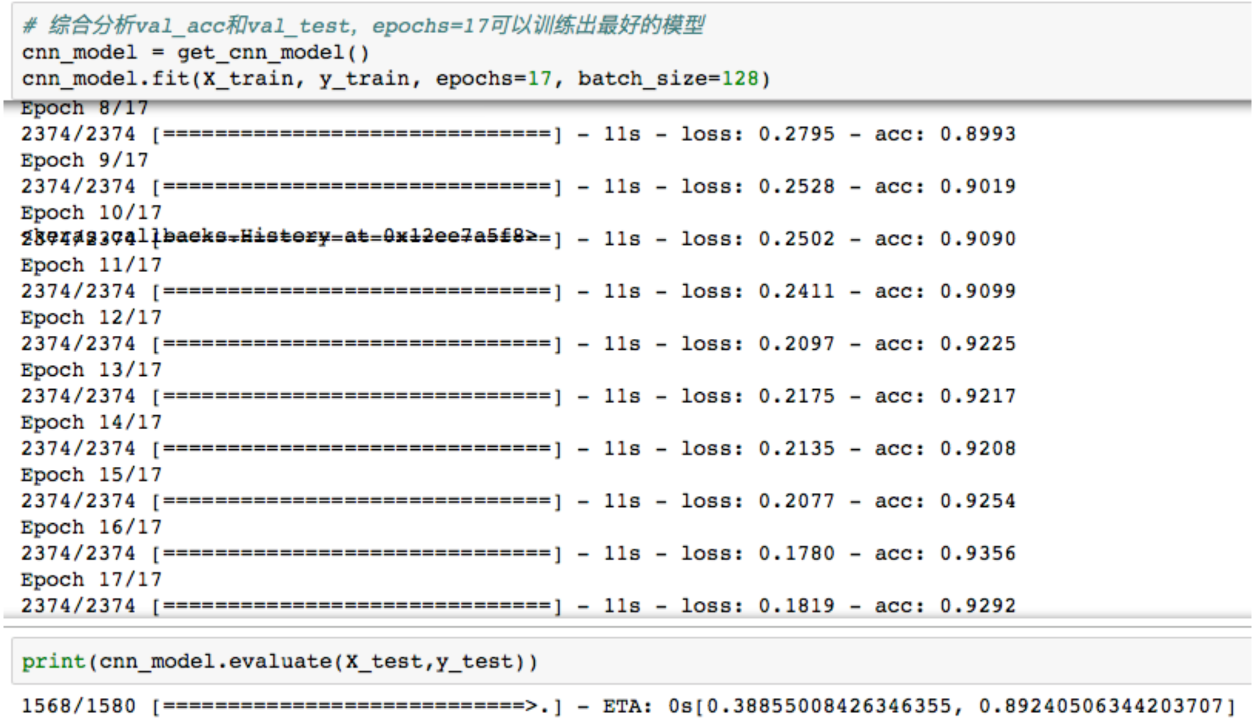
```
y_pred_7 = rf_clf.predict(X_test)
```

```
from sklearn import metrics

metrics.accuracy_score(y_test, y_pred_7)
```

```
0.80696202531645567
```

卷积神经网络



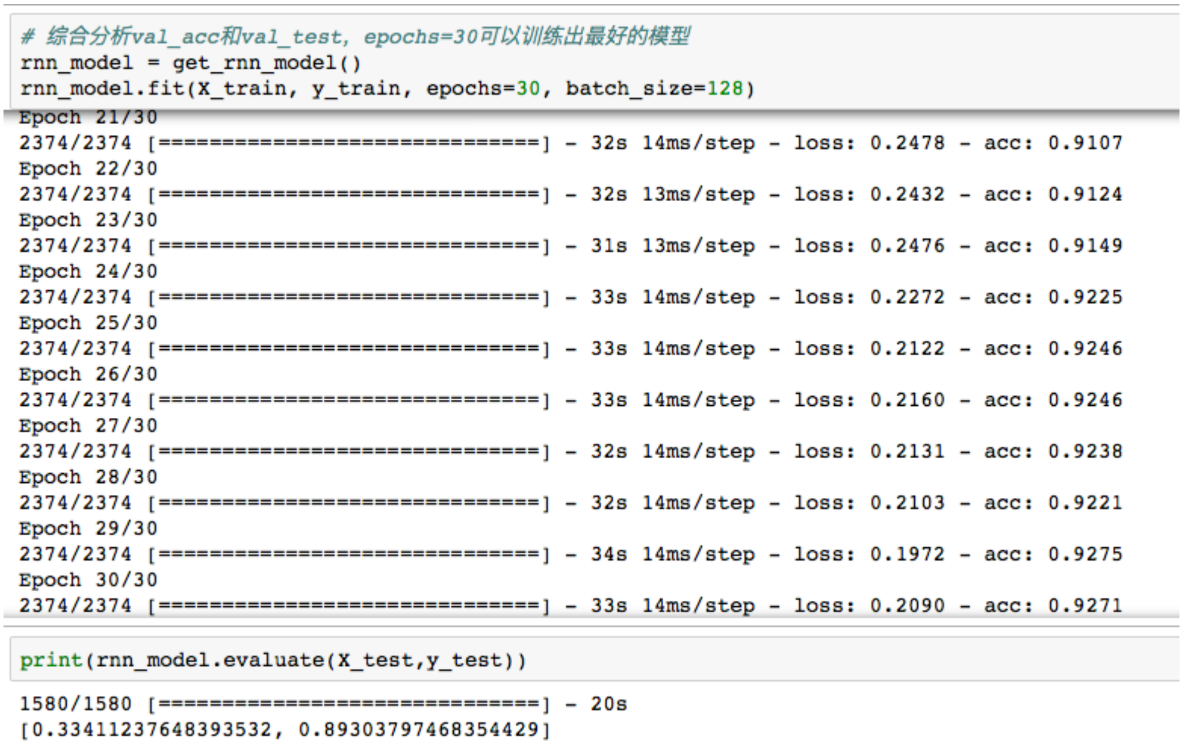
```
# 综合分析val_acc和val_test, epochs=17可以训练出最好的模型
cnn_model = get_cnn_model()
cnn_model.fit(X_train, y_train, epochs=17, batch_size=128)
```

```
Epoch 8/17
2374/2374 [==============================] - 11s - loss: 0.2795 - acc: 0.8993
Epoch 9/17
2374/2374 [==============================] - 11s - loss: 0.2528 - acc: 0.9019
Epoch 10/17
2374/2374 [==============================] - 11s - loss: 0.2502 - acc: 0.9090
<keras.callbacks.History at 0x12ee7a5f8>
Epoch 11/17
2374/2374 [==============================] - 11s - loss: 0.2411 - acc: 0.9099
Epoch 12/17
2374/2374 [==============================] - 11s - loss: 0.2097 - acc: 0.9225
Epoch 13/17
2374/2374 [==============================] - 11s - loss: 0.2175 - acc: 0.9217
Epoch 14/17
2374/2374 [==============================] - 11s - loss: 0.2135 - acc: 0.9208
Epoch 15/17
2374/2374 [==============================] - 11s - loss: 0.2077 - acc: 0.9254
Epoch 16/17
2374/2374 [==============================] - 11s - loss: 0.1780 - acc: 0.9356
Epoch 17/17
2374/2374 [==============================] - 11s - loss: 0.1819 - acc: 0.9292
```

```
print(cnn_model.evaluate(X_test,y_test))
```

```
1568/1580 [============================>.] - ETA: 0s[0.38855008426346355, 0.89240506344203707]
```

卷积神经网络



```
# 综合分析val_acc和val_test, epochs=30可以训练出最好的模型
rnn_model = get_rnn_model()
rnn_model.fit(X_train, y_train, epochs=30, batch_size=128)
```

```
Epoch 21/30
2374/2374 [==============================] - 32s 14ms/step - loss: 0.2478 - acc: 0.9107
Epoch 22/30
2374/2374 [==============================] - 32s 13ms/step - loss: 0.2432 - acc: 0.9124
Epoch 23/30
2374/2374 [==============================] - 31s 13ms/step - loss: 0.2476 - acc: 0.9149
Epoch 24/30
2374/2374 [==============================] - 33s 14ms/step - loss: 0.2272 - acc: 0.9225
Epoch 25/30
2374/2374 [==============================] - 33s 14ms/step - loss: 0.2122 - acc: 0.9246
Epoch 26/30
2374/2374 [==============================] - 33s 14ms/step - loss: 0.2160 - acc: 0.9246
Epoch 27/30
2374/2374 [==============================] - 32s 14ms/step - loss: 0.2131 - acc: 0.9238
Epoch 28/30
2374/2374 [==============================] - 32s 14ms/step - loss: 0.2103 - acc: 0.9221
Epoch 29/30
2374/2374 [==============================] - 34s 14ms/step - loss: 0.1972 - acc: 0.9275
Epoch 30/30
2374/2374 [==============================] - 33s 14ms/step - loss: 0.2090 - acc: 0.9271
```

```
print(rnn_model.evaluate(X_test,y_test))
```

```
1580/1580 [==============================] - 20s
[0.33411237648393532, 0.89303797468354429]
```