Московский государственный технический университет

им. Н.Э. Баумана

(МГТУ им. Н.Э. Баумана)

Радиотехнический факультет (РТ)

Отчёт по лабораторным работам №4-5

По дисциплине

«Технологии машинного обучения»

Проверил:

Преподаватель кафедры ИУ-5

Гапанюк Ю.Е.

Подпись: _____________

«___» _____________ 2020 г.

Выполнил:

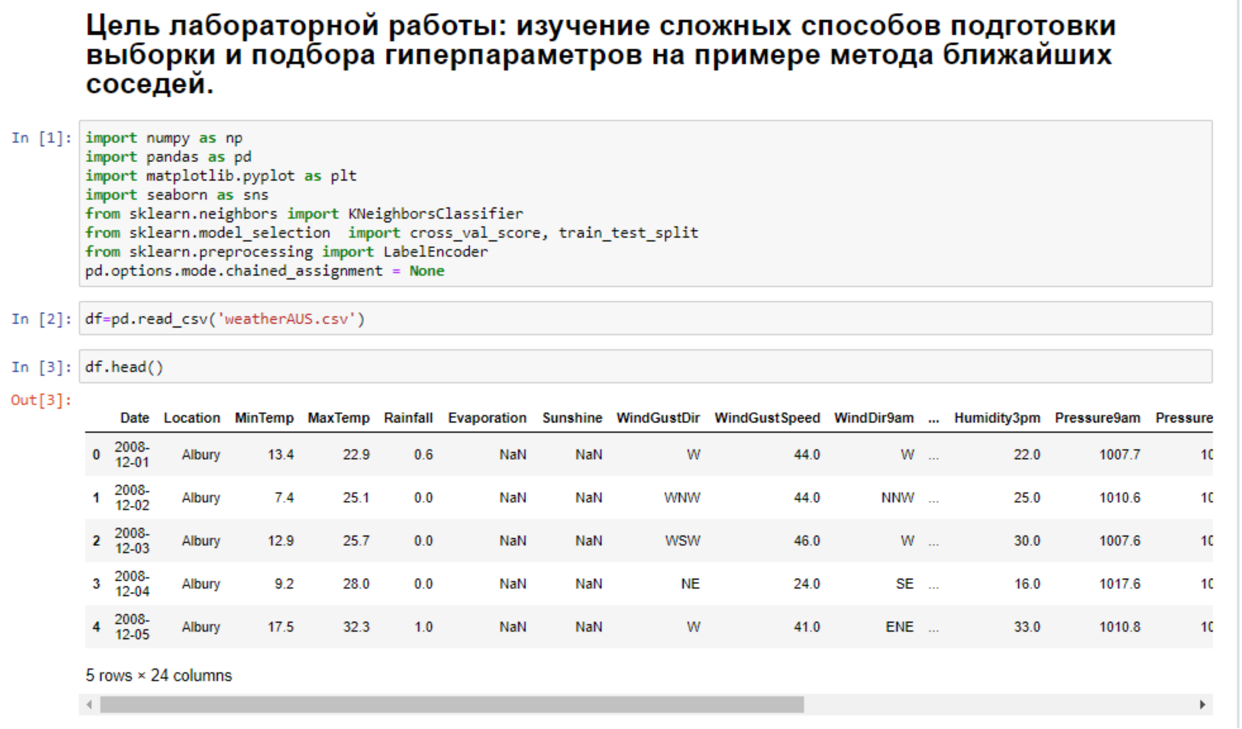
студент группы РТ5-61Б

Ануров Н.С.

Подпись: _____________

«___» _____________ 2020 г.

Москва, 2020



Цель лабораторной работы: изучение сложных способов подготовки выборки и подбора гиперпараметров на примере метода ближайших соседей.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection  import cross_val_score, train_test_split
from sklearn.preprocessing import LabelEncoder
pd.options.mode.chained_assignment = None
```

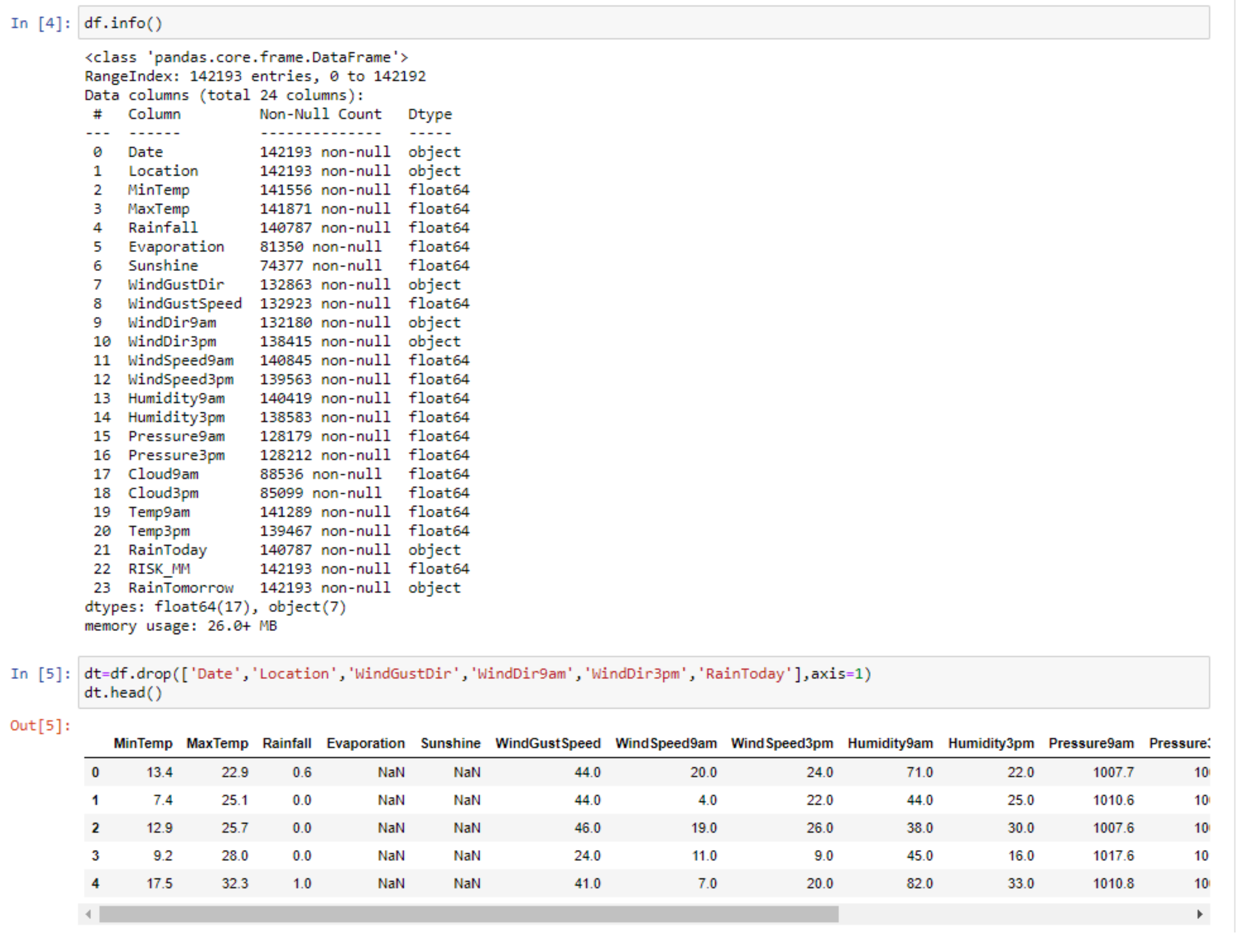
In [2]:
```
df=pd.read_csv('weatherAUS.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity3pm	Pressure9am	Pressure
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	...	22.0	1007.7	10
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	...	25.0	1010.6	10
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	...	30.0	1007.6	10
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	...	16.0	1017.6	10
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	...	33.0	1010.8	10

5 rows × 24 columns



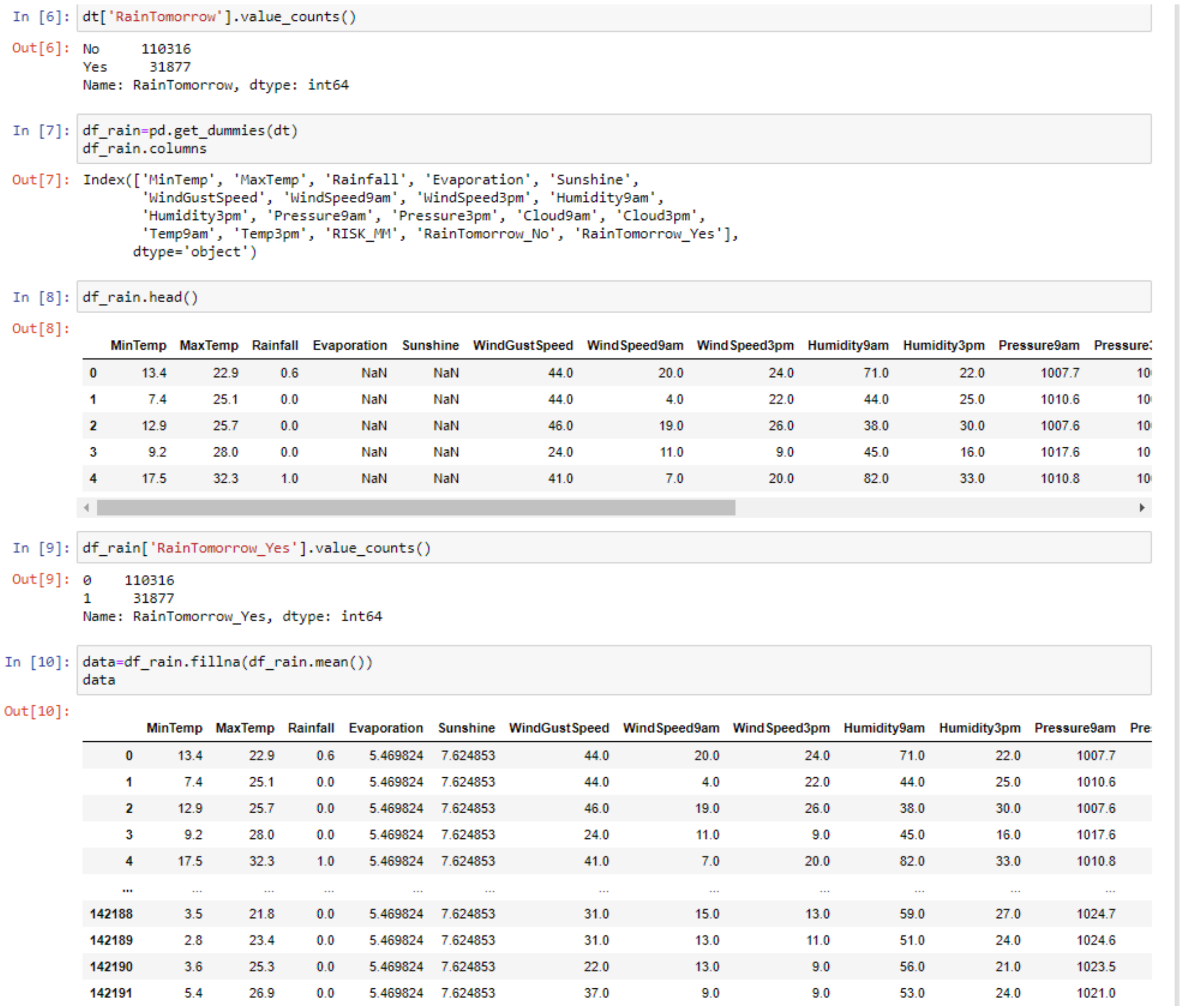
In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 142193 entries, 0 to 142192
Data columns (total 24 columns):
 #   Column         Non-Null Count   Dtype  
---  ------         --------------   -----  
 0   Date           142193 non-null  object 
 1   Location       142193 non-null  object 
 2   MinTemp        141556 non-null  float64
 3   MaxTemp        141871 non-null  float64
 4   Rainfall       140787 non-null  float64
 5   Evaporation    81350 non-null   float64
 6   Sunshine       74377 non-null   float64
 7   WindGustDir    132863 non-null  object 
 8   WindGustSpeed  132923 non-null  float64
 9   WindDir9am     132180 non-null  object 
 10  WindDir3pm     138415 non-null  object 
 11  WindSpeed9am   140845 non-null  float64
 12  WindSpeed3pm   139563 non-null  float64
 13  Humidity9am    140419 non-null  float64
 14  Humidity3pm    138583 non-null  float64
 15  Pressure9am    128179 non-null  float64
 16  Pressure3pm    128212 non-null  float64
 17  Cloud9am       88536 non-null   float64
 18  Cloud3pm       85099 non-null   float64
 19  Temp9am        141289 non-null  float64
 20  Temp3pm        139467 non-null  float64
 21  RainToday      140787 non-null  object 
 22  RISK_MM        142193 non-null  float64
 23  RainTomorrow   142193 non-null  object 
dtypes: float64(17), object(7)
memory usage: 26.0+ MB
```

In [5]:
```
dt=df.drop(['Date','Location','WindGustDir','WindDir9am','WindDir3pm','RainToday'],axis=1)
dt.head()
```

Out[5]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure:
0	13.4	22.9	0.6	NaN	NaN	44.0	20.0	24.0	71.0	22.0	1007.7	10
1	7.4	25.1	0.0	NaN	NaN	44.0	4.0	22.0	44.0	25.0	1010.6	10
2	12.9	25.7	0.0	NaN	NaN	46.0	19.0	26.0	38.0	30.0	1007.6	10
3	9.2	28.0	0.0	NaN	NaN	24.0	11.0	9.0	45.0	16.0	1017.6	10
4	17.5	32.3	1.0	NaN	NaN	41.0	7.0	20.0	82.0	33.0	1010.8	10



In [6]:
```
dt['RainTomorrow'].value_counts()
```

Out[6]:
```
No     110316
Yes     31877
Name: RainTomorrow, dtype: int64
```

In [7]:
```
df_rain=pd.get_dummies(dt)
df_rain.columns
```

Out[7]:
```
Index(['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine',
       'WindGustSpeed', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am',
       'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm',
       'Temp9am', 'Temp3pm', 'RISK_MM', 'RainTomorrow_No', 'RainTomorrow_Yes'],
      dtype='object')
```

In [8]:
```
df_rain.head()
```

Out[8]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure:
0	13.4	22.9	0.6	NaN	NaN	44.0	20.0	24.0	71.0	22.0	1007.7	10
1	7.4	25.1	0.0	NaN	NaN	44.0	4.0	22.0	44.0	25.0	1010.6	10
2	12.9	25.7	0.0	NaN	NaN	46.0	19.0	26.0	38.0	30.0	1007.6	10
3	9.2	28.0	0.0	NaN	NaN	24.0	11.0	9.0	45.0	16.0	1017.6	10
4	17.5	32.3	1.0	NaN	NaN	41.0	7.0	20.0	82.0	33.0	1010.8	10

In [9]:
```
df_rain['RainTomorrow_Yes'].value_counts()
```

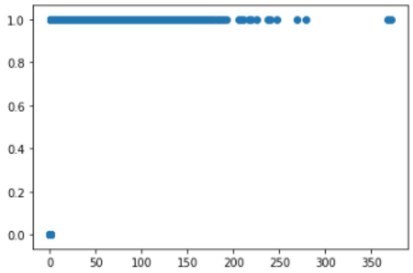
Out[9]:
```
0    110316
1     31877
Name: RainTomorrow_Yes, dtype: int64
```

In [10]:
```
data=df_rain.fillna(df_rain.mean())
data
```

Out[10]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pre
0	13.4	22.9	0.6	5.469824	7.624853	44.0	20.0	24.0	71.0	22.0	1007.7	
1	7.4	25.1	0.0	5.469824	7.624853	44.0	4.0	22.0	44.0	25.0	1010.6	
2	12.9	25.7	0.0	5.469824	7.624853	46.0	19.0	26.0	38.0	30.0	1007.6	
3	9.2	28.0	0.0	5.469824	7.624853	24.0	11.0	9.0	45.0	16.0	1017.6	
4	17.5	32.3	1.0	5.469824	7.624853	41.0	7.0	20.0	82.0	33.0	1010.8	
...	...	...	...	...	...	...	...	...	...	...	...	
142188	3.5	21.8	0.0	5.469824	7.624853	31.0	15.0	13.0	59.0	27.0	1024.7	
142189	2.8	23.4	0.0	5.469824	7.624853	31.0	13.0	11.0	51.0	24.0	1024.6	
142190	3.6	25.3	0.0	5.469824	7.624853	22.0	13.0	9.0	56.0	21.0	1023.5	
142191	5.4	26.9	0.0	5.469824	7.624853	37.0	9.0	9.0	53.0	24.0	1021.0	

In [12]:
```
plt.scatter( x='RISK_MM', y='RainTomorrow_Yes', data=data);
```



In [13]:
```
y=data['RainTomorrow_Yes'].values
X=data.drop(['RainTomorrow_Yes'],axis=1).values
X
```

Out[13]:
```
array([[13.4, 22.9,  0.6, ..., 21.8,  0. ,  1. ],
       [ 7.4, 25.1,  0. , ..., 24.3,  0. ,  1. ],
       [12.9, 25.7,  0. , ..., 23.2,  0. ,  1. ],
       ...,
       [ 3.6, 25.3,  0. , ..., 24.5,  0. ,  1. ],
       [ 5.4, 26.9,  0. , ..., 26.1,  0. ,  1. ],
       [ 7.8, 27. ,  0. , ..., 26. ,  0. ,  1. ]])
```

In [14]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=21)
np.unique(X_train)
```

Out[14]:
```
array([  -8.5,   -8.2,   -8. , ..., 1040.4, 1040.6, 1040.9])
```

In [15]:
```
np.unique(X_test)
```

Out[15]:
```
array([  -7.5,   -7.2,   -7. , ..., 1040.4, 1040.5, 1041. ])
```

In [16]:
```
knn=KNeighborsClassifier(n_neighbors=5)
```

In [17]:
```
knn.fit(X_train,y_train)
```

Out[17]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

In [18]:
```
knn.score(X_test,y_test)
```

Out[18]:
```
0.892892306249707
```

```
In [19]: cross_val_score(knn,X_train,y_train,cv=5)

Out[19]: array([0.89265083, 0.89194756, 0.89877932, 0.89164615, 0.89099312])
```

```
In [20]: np.mean(cross_val_score(knn,X_train,y_train,cv=5))

Out[20]: 0.8932033957904254
```

```
In [21]: from sklearn.model_selection import GridSearchCV
```

```
In [22]: knn_params = {'n_neighbors' : list(range(1,15))}
```

```
In [23]: knn_grid = GridSearchCV(knn,knn_params,cv=5)
```

```
In [24]: knn_grid.fit(X_train,y_train)

Out[24]: GridSearchCV(cv=5, error_score=nan,
                      estimator=KNeighborsClassifier(algorithm='auto', leaf_size=30,
                                                     metric='minkowski',
                                                     metric_params=None, n_jobs=None,
                                                     n_neighbors=5, p=2,
                                                     weights='uniform'),
                      iid='deprecated', n_jobs=None,
                      param_grid={'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
                                                  13, 14]},
                      pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
                      scoring=None, verbose=0)
```

```
In [25]: knn_grid.best_score_, knn_grid.best_params_

Out[25]: (0.8940473200381774, {'n_neighbors': 7})
```

Лабораторная работа №5

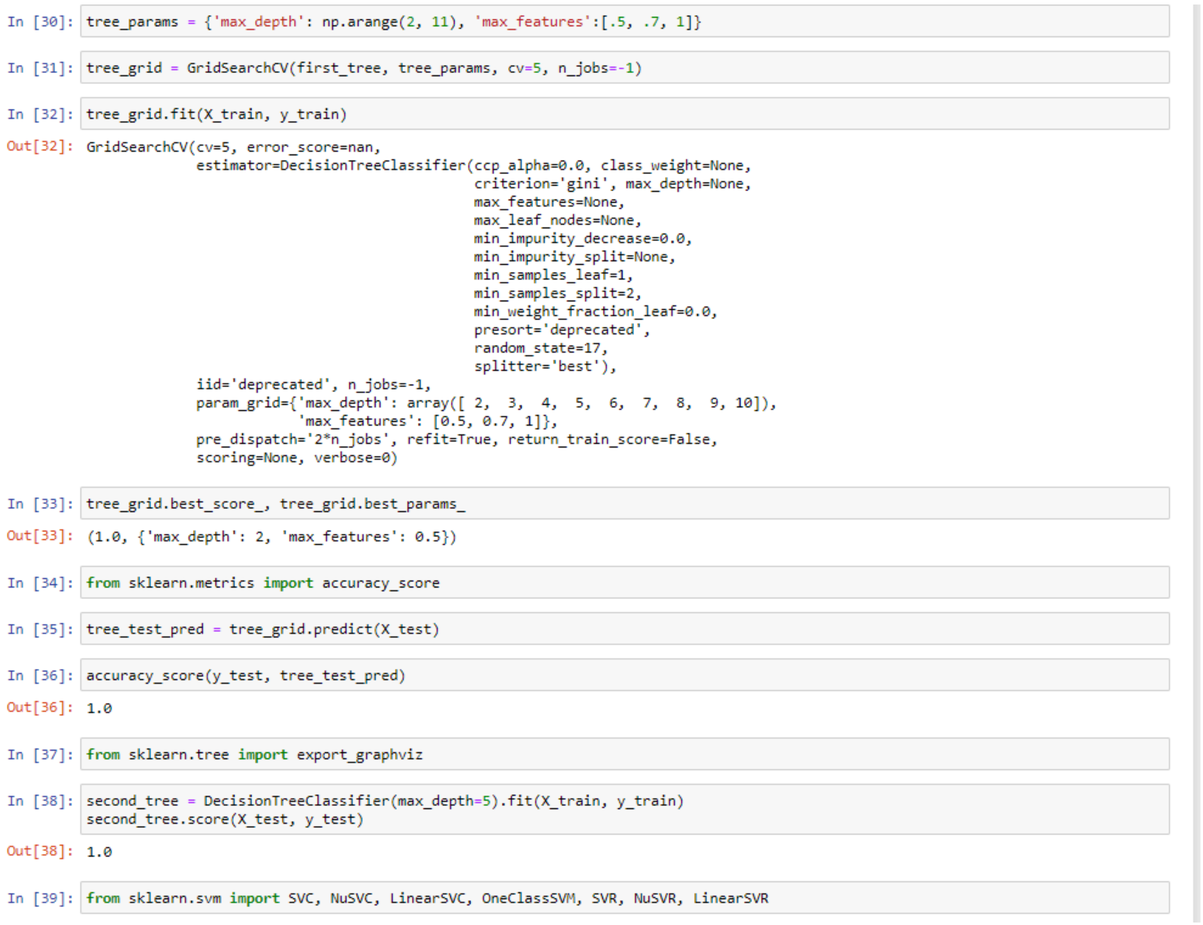
Цель лабораторной работы: изучение линейных моделей, SVM и деревьев решений.

```
In [26]: from sklearn.tree import DecisionTreeClassifier
         from sklearn.svm import SVC
```

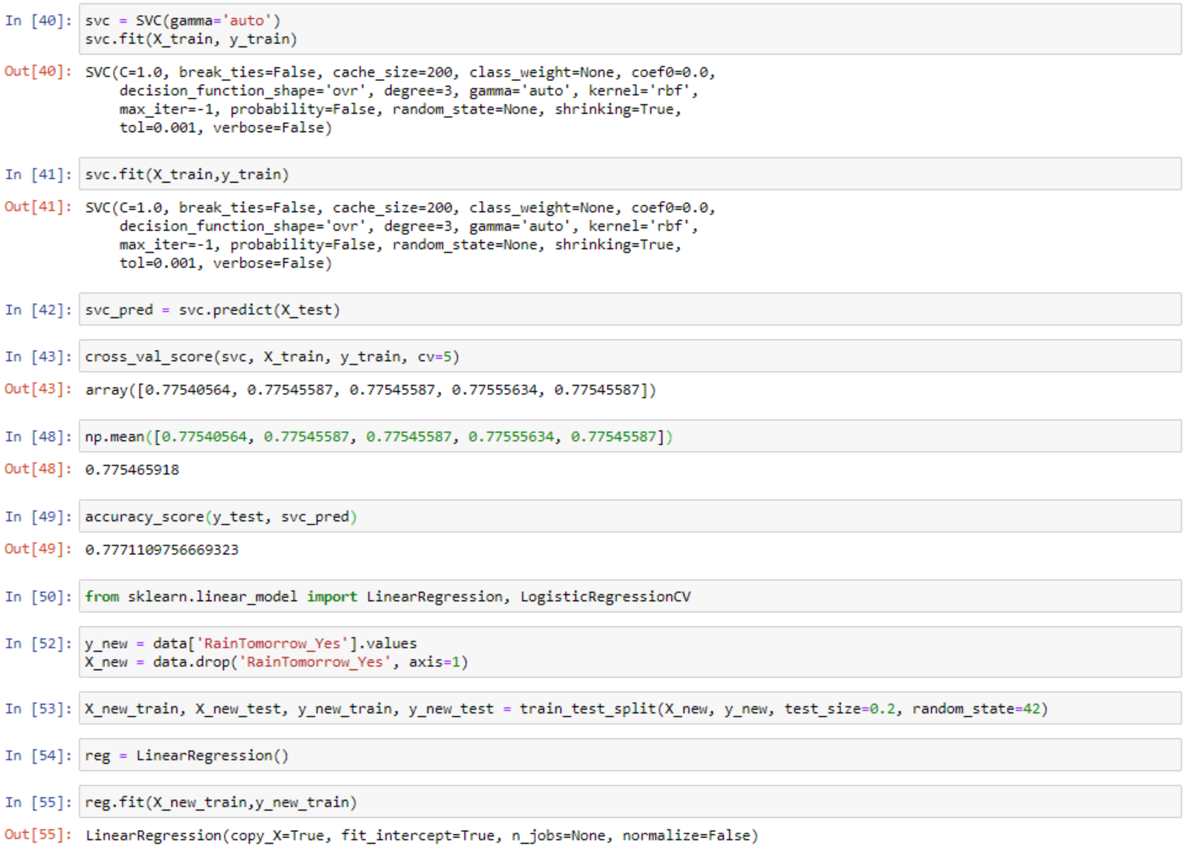
```
In [27]: first_tree = DecisionTreeClassifier(random_state=17)
```

```
In [28]: cross_val_score(first_tree, X_train, y_train, cv=5)

Out[28]: array([1., 1., 1., 1., 1.])
```

```
In [29]: np.mean(cross_val_score(first_tree, X_train, y_train, cv=5))

Out[29]: 1.0
```



```
In [30]: tree_params = {'max_depth': np.arange(2, 11), 'max_features':[.5, .7, 1]}

In [31]: tree_grid = GridSearchCV(first_tree, tree_params, cv=5, n_jobs=-1)

In [32]: tree_grid.fit(X_train, y_train)

Out[32]: GridSearchCV(cv=5, error_score=nan,
                      estimator=DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None,
                                                       criterion='gini', max_depth=None,
                                                       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,
                                                       presort='deprecated',
                                                       random_state=17,
                                                       splitter='best'),
                      iid='deprecated', n_jobs=-1,
                      param_grid={'max_depth': array([ 2,  3,  4,  5,  6,  7,  8,  9, 10]),
                                  'max_features': [0.5, 0.7, 1]},
                      pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
                      scoring=None, verbose=0)

In [33]: tree_grid.best_score_, tree_grid.best_params_

Out[33]: (1.0, {'max_depth': 2, 'max_features': 0.5})

In [34]: from sklearn.metrics import accuracy_score

In [35]: tree_test_pred = tree_grid.predict(X_test)

In [36]: accuracy_score(y_test, tree_test_pred)

Out[36]: 1.0

In [37]: from sklearn.tree import export_graphviz

In [38]: second_tree = DecisionTreeClassifier(max_depth=5).fit(X_train, y_train)
         second_tree.score(X_test, y_test)

Out[38]: 1.0

In [39]: from sklearn.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR, NuSVR, LinearSVR
```



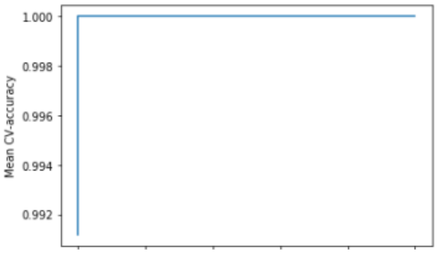
```
In [40]: svc = SVC(gamma='auto')
         svc.fit(X_train, y_train)

Out[40]: SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
             decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
             max_iter=-1, probability=False, random_state=None, shrinking=True,
             tol=0.001, verbose=False)

In [41]: svc.fit(X_train,y_train)

Out[41]: SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
             decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
             max_iter=-1, probability=False, random_state=None, shrinking=True,
             tol=0.001, verbose=False)

In [42]: svc_pred = svc.predict(X_test)

In [43]: cross_val_score(svc, X_train, y_train, cv=5)

Out[43]: array([0.77540564, 0.77545587, 0.77545587, 0.77555634, 0.77545587])

In [48]: np.mean([0.77540564, 0.77545587, 0.77545587, 0.77555634, 0.77545587])

Out[48]: 0.775465918

In [49]: accuracy_score(y_test, svc_pred)

Out[49]: 0.7771109756669323

In [50]: from sklearn.linear_model import LinearRegression, LogisticRegressionCV

In [52]: y_new = data['RainTomorrow_Yes'].values
         X_new = data.drop('RainTomorrow_Yes', axis=1)

In [53]: X_new_train, X_new_test, y_new_train, y_new_test = train_test_split(X_new, y_new, test_size=0.2, random_state=42)

In [54]: reg = LinearRegression()

In [55]: reg.fit(X_new_train,y_new_train)

Out[55]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [56]:
```
from sklearn.model_selection import StratifiedKFold
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=17)

c_values = np.logspace(-2, 3, 500)

logit_searcher = LogisticRegressionCV(Cs=c_values, cv=skf, verbose=1, n_jobs=-1)
logit_searcher.fit(X_new, y_new)
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done   2 out of   5 | elapsed:  9.6min remaining: 14.4min
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed: 10.1min finished
C:\Users\Nikita Anurov\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:940: ConvergenceWarning: lbfgs failed to c
onverge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[56]:
```
LogisticRegressionCV(Cs=array([1.00000000e-02, 1.02334021e-02, 1.04722519e-02, 1.07166765e-02,
       1.09668060e-02, 1.12227736e-02, 1.14847155e-02, 1.17527712e-02,
       1.20270833e-02, 1.23077980e-02, 1.25950646e-02, 1.28890361e-02,
       1.31898690e-02, 1.34977233e-02, 1.38127630e-02, 1.41351558e-02,
       1.44650734e-02, 1.48026913e-02, 1.51481892e-02, 1.55017512e-02,
       1.58635653e-02, 1.62...
       8.50863158e+02, 8.70722485e+02, 8.91045332e+02, 9.11842520e+02,
       9.33125118e+02, 9.54904456e+02, 9.77192128e+02, 1.00000000e+03]),
                     class_weight=None,
                     cv=StratifiedKFold(n_splits=5, random_state=17, shuffle=True),
                     dual=False, fit_intercept=True, intercept_scaling=1.0,
                     l1_ratios=None, max_iter=100, multi_class='auto',
                     n_jobs=-1, penalty='l2', random_state=None, refit=True,
                     scoring=None, solver='lbfgs', tol=0.0001, verbose=1)
```

In [57]:
```
plt.plot(c_values, np.mean(logit_searcher.scores_[1], axis=0))
plt.xlabel('C')
plt.ylabel('Mean CV-accuracy');
```


Mean CV-accuracy