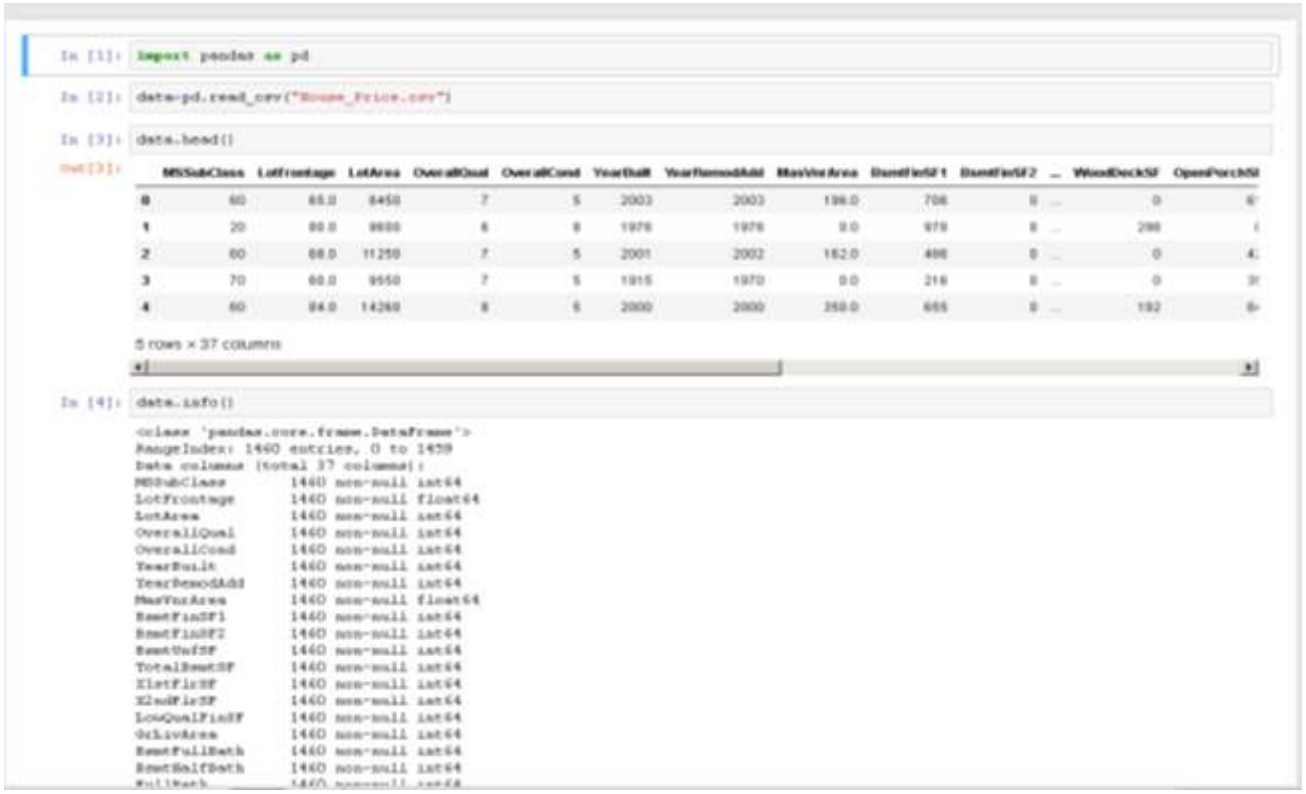
Лабораторное занятие 4. Работа с библиотекой sklearn.tree import DecisionTreeClassifier в python



```
In [1]: import pandas as pd

In [2]: data=pd.read_csv("House_Price.csv")

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

Out[3]:

MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	...	WoodDeckSF	OpenPorchSF
0	60	65.0	8450	7	5	2003	2003	196.0	706	0	...	0	6
1	20	80.0	9600	6	8	1976	1976	0.0	978	0	...	298	
2	60	68.0	11250	7	5	2001	2002	162.0	486	0	...	0	4
3	70	60.0	9550	7	5	1915	1970	0.0	216	0	...	0	3
4	60	84.0	14260	8	5	2000	2000	350.0	655	0	...	192	6

5 rows × 37 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 37 columns):
MSSubClass       1460 non-null int64
LotFrontage      1460 non-null float64
LotArea          1460 non-null int64
OverallQual      1460 non-null int64
OverallCond      1460 non-null int64
YearBuilt        1460 non-null int64
YearRemodAdd     1460 non-null int64
MasVnrArea       1460 non-null float64
BsmtFinSF1       1460 non-null int64
BsmtFinSF2       1460 non-null int64
BsmtUnfSF        1460 non-null int64
TotalBsmtSF      1460 non-null int64
1stFlrSF         1460 non-null int64
2ndFlrSF         1460 non-null int64
LowQualFinSF     1460 non-null int64
GrLivArea        1460 non-null int64
BsmtFullBath     1460 non-null int64
BsmtHalfBath     1460 non-null int64
```

```
ScreenPorch      1460 non-null int64
PoolArea         1460 non-null int64
MiscVal          1460 non-null int64
MoSold           1460 non-null int64
YrSold           1460 non-null int64
SalePrice        1460 non-null int64
dtypes: float64(3), int64(34)
memory usage: 422.1 KB

In [7]: y=data['SalePrice'].astype('int')
        #Преобразуем тип int

In [9]: X=data.drop(['SalePrice'], axis=1)

In [10]: X.shape, y.shape

Out[10]: ((1460, 36), (1460,))

In [11]: from sklearn.model_selection import train_test_split

In [12]: X_train, X_valid, y_train, y_valid = train_test_split(X, y,
                                                    test_size=0.3)

In [14]: X_train.shape, X_valid.shape

Out[14]: ((1022, 36), (438, 36))

In [57]: from sklearn.tree import DecisionTreeClassifier
         tree = DecisionTreeClassifier(max_depth=5, min_samples_leaf=1, max_features=36, criterion="gini")

In [58]: tree.fit(X_train, y_train)
         Y_pred = tree.predict(X_valid)

In [59]: from sklearn import metrics
         print("recall = " + str(metrics.recall_score(y_valid, Y_pred)))
         print("precision = " + str(metrics.precision_score(y_valid, Y_pred)))
         print("f1 = " + str(metrics.f1_score(y_valid, Y_pred)))
         print("accuracy = " + str(metrics.accuracy_score(y_valid, Y_pred)))

         recall = 0.8650306748466258
         precision = 0.8703703703703703
         f1 = 0.8676923076923078
```

```
accuracy = 0.9018164840182648
```

In [54]:
```
import numpy as np
from sklearn.model_selection import cross_val_score
np.mean(cross_val_score(tree, X_train, y_train, cv=40))
```

Out[54]:
```
0.8982916666666668
```

In [55]:
```
cross_val_score(tree, X_train, y_train,cv=40)
```

Out[55]:
```
array([0.88461538, 0.88461538, 0.92307692, 0.92307692, 0.84615385,
       0.96153846, 0.96153846, 0.88461538, 0.76923077, 0.96153846,
       0.88461538, 0.92307692, 0.96153846, 1.        , 0.88461538,
       0.84615385, 0.96153846, 0.96153846, 0.88461538, 0.73076923,
       0.88461538, 0.84615385, 0.92307692, 0.96153846, 0.92307692,
       0.88      , 0.92      , 0.8       , 0.92      , 0.88      ,
       0.88      , 0.8       , 0.88      , 0.92      , 0.96      ,
       0.96      , 0.88      , 1.        , 0.975     , 0.91666667])
```

In [56]:
```
np.mean(cross_val_score(tree, X_train, y_train,cv=40))
```

Out[56]:
```
0.9002147435897436
```

In [64]:
```
from sklearn.model_selection import GridSearchCV

tuned_parametersTree = [{"max_depth": np.arange(1, 15),
                         "max_features": [1, 36],
                         "criterion": ["entropy","gini"],
                         "min_samples_leaf" : [1,10]}]
gstree = GridSearchCV(tree, tuned_parametersTree)
```

In [65]:
```
gstree.fit(X_train, y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py:2053: FutureWarning: You should specify a valu
e for 'cv' instead of relying on the default value. The default value will change from 3 to 5 in version 0.22.
  warnings.warn(CV_WARNING, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py:841: DeprecationWarning: The default of the `
iid` parameter will change from True to False in version 0.22 and will be removed in 0.24. This will change numeric results
when test-set sizes are unequal.
  DeprecationWarning)
```

Out[65]:
```
GridSearchCV(cv='warn', error_score='raise-deprecating',
       estimator=DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=5,
            max_features=36, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
```


In [64]:
```
from sklearn.model_selection import GridSearchCV

tuned_parametersTree = [{"max_depth": np.arange(1, 15),
                         "max_features": [1, 36],
                         "criterion": ["entropy","gini"],
                         "min_samples_leaf" : [1,10]}]
gstree = GridSearchCV(tree, tuned_parametersTree)
```

In [65]:
```
gstree.fit(X_train, y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py:2053: FutureWarning: You should specify a valu
e for 'cv' instead of relying on the default value. The default value will change from 3 to 5 in version 0.22.
  warnings.warn(CV_WARNING, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py:841: DeprecationWarning: The default of the `
iid` parameter will change from True to False in version 0.22 and will be removed in 0.24. This will change numeric results
when test-set sizes are unequal.
  DeprecationWarning)
```

Out[65]:
```
GridSearchCV(cv='warn', error_score='raise-deprecating',
       estimator=DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=5,
            max_features=36, 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=False, random_state=None,
            splitter='best'),
       fit_params=None, iid='warn', n_jobs=None,
       param_grid=[{'max_depth': array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14]), 'max_features': [1, 36],
'criterion': ['entropy', 'gini'], 'min_samples_leaf': [1, 10]}],
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0)
```


In [66]:
```
gstree.best_params_
```

Out[66]:
```
{'criterion': 'gini',
 'max_depth': 5,
 'max_features': 36,
 'min_samples_leaf': 1}
```

Задача 1. Взять другой датасеть для него разработаь алгоритм дереворешений