Московский государственный технический университет им. Н. Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

Отчет по

лабораторной работе № 3 по курсу

«Введение в машинное обучение»

Исполнитель: ИУ5-41, Черепанов Е.

Преподаватель: Гапанюк Ю.

Москва, 2018г.

Задание.

Необходимо решить задачу предсказания стоимости дома в зависимости от его характеристик.

1. Провести предподготовку данных.
2. Разделить данные.
3. Обучить модель из sklearn.
4. Реализовать линейную регрессию.
5. Эксперименты с моделью.

Код и результаты.

In [5]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy import stats
import warnings
warnings.filterwarnings('ignore')
from sklearn.cross_validation import train_test_split
from sklearn import linear_model
from sklearn.metrics import mean_absolute_error, r2_score
%matplotlib inline
```

In [6]:
```
data = pd.read_csv('./ML_d/train.csv')
```

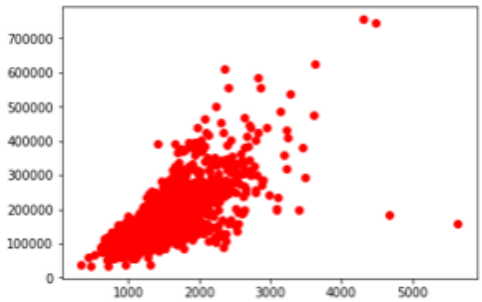
In [7]:
```
data
```

Out[7]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscV
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	
5	6	50	RL	85.0	14115	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	Shed	70
6	7	20	RL	75.0	10084	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	
7	8	60	RL	NaN	10382	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	Shed	35
8	9	50	RM	51.0	6120	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	
9	10	190	RL	50.0	7420	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	
10	11	20	RL	70.0	11200	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	

In [8]:
```
#распределение площади домов и цен
plt.plot(data['GrLivArea'], data['SalePrice'], 'ro')
```

Out[8]:
```
[<matplotlib.lines.Line2D at 0x167565ab780>]
```



In [9]:
```
def cleaning(data):
    categorical_columns = [c for c in data.columns if data[c].dtype.name == 'objekt']
    numerical_columns = [c for c in data.columns if (data[c].dtype.name != 'objekt' and c != 'SalePrice')]
    answer_column = [c for c in data.columns if c == 'SalePrice']
    #заполнение пустых количественных медианным значением
    data = data.fillna(data.median(axis = 0), axis = 0)
    #заполнение пустых категориальных самым частым значением по признаку
    data_describe = data.describe(include = [object]) #получение сводной информации по таблице
    for c in categorical_columns:
        data[c] = data[c].fillna(data_describe[c]['top'])
        # fillna() - метод для замены отсутствующих значений на числовые
    #проведение векторизации - перевод категориальных признаков в количественные
    binary_columns = [c for c in categorical_columns if data_describe[c]['unique'] == 2] #бинарные
    nonbinary_columns = [c for c in categorical_columns if data_describe[c]['unique'] > 2] #небинарные
    for c in binary_columns:
        top = data_describe[c]['top']
        top_items = data[c] == top
        data.loc[top_items, c] = 0
        data.loc[np.logical_not(top_items), c] = 1
        data_nonbinary = pd.get_dummies(data[nonbinary_columns]) #возврат нового столбца для каждого элемента
    #проведение нормализации количественных признаков
    data_numerical = data[numerical_columns]
    data_numerical = (data_numerical - data_numerical.mean()) / data_numerical.std()
    data_answer = data[answer_column] #нормализация не нужна
    #соединение результатов в таблицу
    data = pd.concat((data_numerical, data[binary_columns], data_nonbinary, data_answer), axis = 1)
    data = pd.DataFrame(data, dtype = float)
    return data
```

In [10]:
```
data.corr()
```

Out[10]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	...	WoodDecl
Id	1.000000	0.011156	-0.010601	-0.033226	-0.028365	0.012609	-0.012713	-0.021998	-0.050298	-0.005024	...	-0.029
MSSubClass	0.011156	1.000000	-0.386347	-0.139781	0.032628	-0.059316	0.027850	0.040581	0.022936	-0.069836	...	-0.012
LotFrontage	-0.010601	-0.386347	1.000000	0.426095	0.251646	-0.059213	0.123349	0.088866	0.193458	0.233633	...	0.088
LotArea	-0.033226	-0.139781	0.426095	1.000000	0.105806	-0.005636	0.014228	0.013788	0.104160	0.214103	...	0.171
OverallQual	-0.028365	0.032628	0.251646	0.105806	1.000000	-0.091932	0.572323	0.550684	0.411876	0.239666	...	0.238
OverallCond	0.012609	-0.059316	-0.059213	-0.005636	-0.091932	1.000000	-0.375983	0.073741	-0.128101	-0.046231	...	-0.003
YearBuilt	-0.012713	0.027850	0.123349	0.014228	0.572323	-0.375983	1.000000	0.592855	0.315707	0.249503	...	0.224
YearRemodAdd	-0.021998	0.040581	0.088866	0.013788	0.550684	0.073741	0.592855	1.000000	0.179618	0.128451	...	0.205
MasVnrArea	-0.050298	0.022936	0.193458	0.104160	0.411876	-0.128101	0.315707	0.179618	1.000000	0.264736	...	0.159
BsmtFinSF1	-0.005024	-0.069836	0.233633	0.214103	0.239666	-0.046231	0.249503	0.128451	0.264736	1.000000	...	0.204
BsmtFinSF2	-0.005968	-0.065649	0.049900	0.111170	-0.059119	0.040229	-0.049107	-0.067759	-0.072319	-0.050117	...	0.067

In [11]:
```
data.corr()['SalePrice'].abs().sort_values(ascending = False)
```

Out[11]:
```
SalePrice       1.000000
OverallQual     0.790982
GrLivArea       0.708624
GarageCars      0.640409
GarageArea      0.623431
TotalBsmtSF     0.613581
1stFlrSF        0.605852
FullBath        0.560664
TotRmsAbvGrd    0.533723
YearBuilt       0.522897
YearRemodAdd    0.507101
GarageYrBlt     0.486362
MasVnrArea      0.477493
Fireplaces      0.466929
BsmtFinSF1      0.386420
LotFrontage     0.351799
WoodDeckSF      0.324413
2ndFlrSF        0.319334
OpenPorchSF     0.315856
HalfBath        0.284108
```

In [12]:
```
data['GrLivArea'].corr(data['TotalBsmtSF'])
```

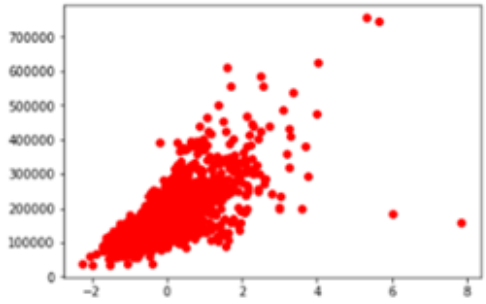
Out[12]:
```
0.45486820254790294
```

In [43]:
```
data['GrLivArea'].corr(data['TotalBsmtSF'])
```

Out[43]:
```
0.45486820254790294
```

In [47]:
```
data = data[['OverallQual','GrLivArea','GarageCars','TotalBsmtSF','ExterQual_TA','FullBath','BsmtQual_Ex',
'TotRmsAbvGrd','YearBuilt','KitchenQual_TA','GarageFinish_Unf','KitchenQual_Ex','SalePrice']]
plt.plot(data['GrLivArea'], data['SalePrice'], 'ro')
```

Out[47]:
```
[<matplotlib.lines.Line2D at 0x25a68390cf8>]
```



In [49]:
```
# Разделить данные

x = data.drop(('SalePrice'), axis=1) # входные фичи

y = data['SalePrice'] # ответ

x_train, x_valid, y_train, y_valid = train_test_split(x, y, test_size = 0.25, random_state = 11)

#Обучить модель из sklearn, реализовать линейную регрессию

regr = linear_model.LinearRegression(fit_intercept=True)

regr.fit(x_train, y_train)

y_valid_predict = regr.predict(x_valid)

print('Коэффициенты: \n', regr.coef_)

print("MAE(средний модуль ошибки): %.2f" % mean_absolute_error(y_valid, y_valid_predict))

print('Оценка дисперсии: %.2f' % r2_score(y_valid, y_valid_predict))
```

```
Коэффициенты:
 [ 20040.67868727  21282.06721516  10568.03838394   8025.47815134
  -5591.26260915  -1630.22122407  33991.18852775   2996.11055473
   6258.80121185  -9984.24646655  -7607.93499391  29703.13546797]
MAE(средний модуль ошибки): 21247.26
Оценка дисперсии: 0.83
```

In [51]:
```
#прогон модели по тестовой выборке

x_test = pd.read_csv('./KaggleLab3/test.csv')

y_test = pd.read_csv('./KaggleLab3/sample_submission.csv')

x_test = cleaning(x_test)

x_test = x_test[['OverallQual','GrLivArea','GarageCars','TotalBsmtSF','ExterQual_TA','FullBath','BsmtQual_Ex','TotRmsAbvGrd','Yea

y_test = y_test[['SalePrice']]

#предсказание

y_test_predict = regr.predict(x_test)

print('Коэффициенты: \n', regr.coef_)

print("MAE: %.2f" % mean_absolute_error(y_test, y_test_predict))

print('Оценка дисперсии: %.2f' % r2_score(y_test, y_test_predict))
```

```
Коэффициенты:
 [ 20040.67868727  21282.06721516  10568.03838394   8025.47815134
  -5591.26260915  -1630.22122407  33991.18852775   2996.11055473
   6258.80121185  -9984.24646655  -7607.93499391  29703.13546797]
MAE: 52785.16
Оценка дисперсии: -15.53
```