Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет «Высшая школа экономики»

Департамент прикладной математики
бакалавриат

О Т Ч Е Т
по проектной работе

Построение оптимальной модели размещения отделения банка

Выполнили студенты гр.БПМ153
Леонов Павел
Загуменнов Филипп
Грошев Александр
Хачатрян Армен
Паньков Кирилл
Евсеев Дмитрий

(подпись)

Руководитель проекта:

_____
(должность, ФИО руководителя проекта)

_____ _____
(оценка) *(подпись)*

_____
(дата)

_____ **2018**

Содержание

1. Общее описание проекта

Основной целью проекта было получение модели, которая предсказывает для конкретного отделения степень его эффективности.

Проект осуществлялся в два этапа: сначала были собраны и обработаны данные, затем они анализировались. На основе этого анализа была реализована с помощью готовых решений языка программирования Python, в частности библиотеки scikit-learn, модель.

В связи со спецификой необходимой информации, сбор данных осуществлялся вручную и представлял из себя достаточно трудоемкий процесс. По этой причине в нем принимали участие 4 человека.

2. Распределение ролей

- *Леонов Павел*

Менеджер проекта; анализ данных; оформление отчета.

- *Грошев Александр*

Анализ полученных данных; консультация участников проекта по технической стороне анализа; поиск оптимальных решений для анализа.

- *Филипп Загуменнов*

Руководство над процессом сбора данных; сбор данных в Москве; консультация участников проекта по вопросам урбанистки и сбора данных; решения для эффективного сбора данных; анализ результатов.

- *Хачатрян Аремен*

Сбор данных в Санкт-Петербурге; поиск оптимальных решений для сбора данных и организации проекта; анализ данных.

- *Паньков Кирилл*

Сбор данных в Самаре и в Москве; анализ полученных результатов.

- *Евсеев Дмитрий*

Сбор данных в Самаре и в Москве; первичная обработка информации; анализ полученных результатов.

2. Сбор данных

Сбор данных осуществлялся в трех городах: Москве, Самаре и Санкт-Петербурге. В каждом из городов участники проекта лично обходили около 100 отделений.

В конкретном отделении собиралась следующая информация:

1) расстояние до метро (км);
2) расстояние до остановок общественного транспорта (км);
3) расстояние до ближайшей значительной проезжей части (км);
4) площадь отделения (м2);
5) полезность отделения;
6) рейтинг отделения.

В нашей задаче были взяты две функции полезности: целочисленная (6 пункт - рейтинг) и вещественная (5 пункт - полезность), которые имеют следующий вид:

$$U(Q_1, Q_2, Q_3, Q_4, Q_5) = 0,5 \cdot (0,3 \cdot Q_1 + 0,2 \cdot Q_2 + 0,5 \cdot Q_3) + 0,5 \cdot (0,5 \cdot Q_4 + 0,5 \cdot Q_5)$$

- Q_1 - кол-во людей в будний день (9:00)
- Q_2 - кол-во людей в будний день (13:00)
- Q_3 - кол-во людей в будний день (18:00)
- Q_4 - кол-во людей в выходной день (12:00)
- Q_5 - кол-во людей в выходной день (17:00)

$$R(U) = \begin{cases} 1, & U \text{ от } 10 \text{ до } 13.4 \\ 2, & U \text{ от } 13.5 \text{ до } 19.4 \\ 3, & U \text{ от } 19.5 \text{ до } 23.4 \\ 4, & U \text{ от } 23.5 \text{ до } 26.4 \\ 5, & U \text{ от } 26.4 \text{ до } 30 \end{cases}$$

Учитывались 3 временных интервала 9:00; 13:00; 18:00 в будний день и 2 временных интервала 12:00; 17:00 в выходной день. Замеры производились по 15 мин.
По результатам замеров было получено количество людей в каждом из трёх временных интервалов и подсчитана полезности отделения. По результатам полезности – рейтинг.

3. Анализ данных

После того, как данные были собраны и обработаны, их анализировали. Далее приведем анализ, который был реализован на языке Python с использованием библиотеки ***scikit-learn***[1].

Подключим некоторые дополнительные библиотек, которые пригодятся в дальнейшем, в их числе: ***numpy***[2], ***pandas***[3] и ***matplotlib***[4].

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

Обработанные данные лежат в файле Bank_opt.csv, загрузим его в таблицу данных, которая поддерживается библиотекой pandas. Обозначим таблицу символами df1.

```
df1 = pd.read_csv('Bank_opt.csv', index_col =0)
df1.head(5)
```

	metro	transport	road	area	utility	rank
0	1.02	0.34	0.89	198.0	12.5	1.0
1	0.74	0.28	1.46	250.0	11.8	1.0
2	0.80	0.49	0.28	135.0	13.3	1.0
3	0.90	0.24	0.12	119.0	11.4	1.0
4	1.18	0.77	1.26	214.0	12.6	1.0

```
df1.columns
```

```
Index(['metro', 'transport', 'road', 'area', 'utility', 'rank'], dtype='object')
```

Имеется два столбца с ответами: rank (рейтинг) и utility (полезность), которые и будут предсказываться моделями.

---

[1] http://scikit-learn.org/stable/index.html
[2] http://www.numpy.org/
[3] http://pandas.pydata.org/
[4] https://matplotlib.org/

В первом случае ответы целочисленные, то есть возникает задача классификации, во втором случае ответы вещественные, поэтому задача регрессии.

Решим отдельно эти задачи, найдем наиболее эффективные модели, выясним, какие признаки вносят наибольший вклад в предсказываем ответ и сравним полученные результаты.

3.1 Задача классификации

Удалим из таблицы данных столбцы ответов rank и utility, при этом запишем их в отдельные переменные df_ans_1 и df_ans_2 соответственно.

```
df_ans_1 = df1["rank"]
df1.drop('rank', axis=1, inplace=True)
df_ans_2 = df1["utility"]
df1.drop('utility', axis=1, inplace=True)
```

Теперь отмасштабируем наши признаки, чтобы их значения лежали в отрезке [0,1]. Это стандартная процедура при анализе данных, она способствует повышению точности модели. Измененную таблицу обозначим просто df.

```
from sklearn.preprocessing import MinMaxScaler
scal = MinMaxScaler(feature_range=(0,1))
scal.fit(df1)
df = pd.DataFrame(scal.transform(df1))
df.head()
```

	0	1	2	3
0	0.171477	0.316479	0.148687	0.750000
1	0.123939	0.260300	0.244021	1.000000
2	0.134126	0.456929	0.046663	0.447115
3	0.151104	0.222846	0.019903	0.370192
4	0.198642	0.719101	0.210570	0.826923

Разобьём выборку на обучаю (на ней будем обучать модель) и контрольную (на ней будем исследовать точность модели) в отношении 7 к 3.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
train_test_split(df,df_ans_1,test_size=0.3,random_state=42)
```

Для задачи классификации будем использовать три модели: ***RandomForestClassifier***[5] (решающие деревья), ***GaussianNB***[6] (наивный байесовский классификатор) и ***KNeighborsClassifier***[7]. В качестве метрики качества возьмем ***accuracy_score*** [8] – долю правильно предсказанных ответов.

3.1.1 RandomForestClassifier

В основу классификатора положены решающие деревья, которые в комплексе образуют так называемый решающий лес.
Обучим модель, получим предсказание для тестовой выборки и оценим качество.

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(X_train,y_train)
# предсказание
y_pred_rfc=rfc.predict(X_test)
# число правильных ответов к общему числу
accuracy_score (y_true=y_test, y_pred= y_pred_rfc)
```

Для данного классификатора доля правильных ответов получилась равной `0.41509433962264153`

[5] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[6] http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html
[7] http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
[8] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html

3.1.2 GaussianNB

В основу данного классификатор положены вероятностные методы, в частности, теорема Байеса.
Обучим модель, получим предсказание для тестовой выборки и оценим качество.

```
# обучение
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train,y_train)
# предсказание
y_pred_gnb=gnb.predict(X_test)
# число правильных ответов к общему числу
accuracy_score (y_true=y_test, y_pred= y_pred_gnb)
```

Для данного классификатора доля правильных ответов получилась равной `0.27358490566037735`

3.1.3 KNeighborsClassifier

Это метрический классификатор, который при определении класса нового объекта смотрит на его k ближайших соседей. Из самого популярного класса соседей делается вывод о классе нового объекта.
Обучим модель, получим предсказание для тестовой выборки и оценим качество.

```
# обучение
from sklearn.neighbors import KNeighborsClassifier
cl= KNeighborsClassifier (n_neighbors=5)
cl.fit (X_train,y_train)
# предсказание
y_pred_cl=cl.predict(X_test)
# число правильных ответов к общему числу
accuracy_score (y_true=y_test, y_pred= y_pred_cl)
```

Для данного классификатора доля правильных ответов получилась равной `0.3867924528301887`
Итак, из трех выбранных классификаторов самым точным для данной задачи оказался ***RandomForestClassifier***, который, согласно тестовой выборке, будет давать 42 правильных ответа для 100 объектов.

Для дальнейшего анализа зафиксируем максимальные и минимальные значения признаков наших имеющихся объектов еще до нормировки.

```
# максимальные
scal.data_max_
 array([   5.9 ,    1.07,    5.98,  250.  ])


# минимальные
scal.data_min_
 array([  1.00000000e-02,   2.00000000e-03,   1.00000000e-03,
          4.20000000e+01])
```

Чтобы оценить, как конкретный признак влияет на рейтинг, поступим следующим образом: будем изменять некоторый признак при прочих фиксированных средних значениях и посмотрим на изменение предсказываемого рейтинга.

Найдем средние значения всех признаков имеющихся данных до нормировки.

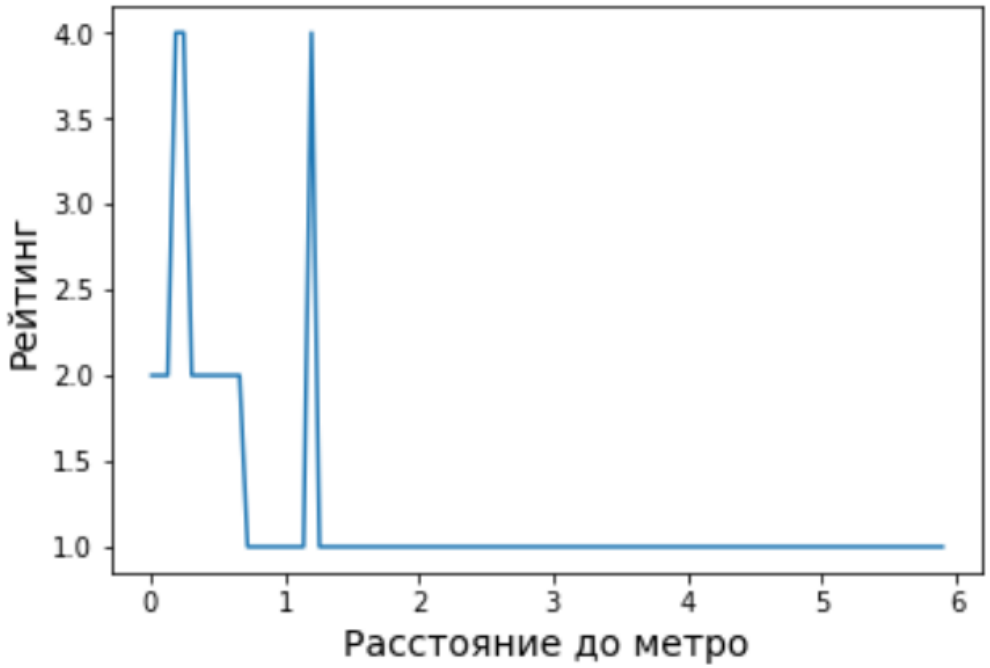
```
m_mean = np.mean(df1['metro'])
t_mean = np.mean(df1['transport'])
r_mean = np.mean(df1['road'])
a_mean = np.mean(df1['area'])
```

3.1.4 Важность признака при классификации

Зафиксируем все признаки, кроме дальности до метро. Будем изменять значения дальности до метро от минимального до максимального, создадим 100 объектов, у которых фиксированы значения всех признаков, кроме дальности до метро. Предскажем

для этих объектов полезность с помощью полученной ранее модели и отобразим информацию на графике.

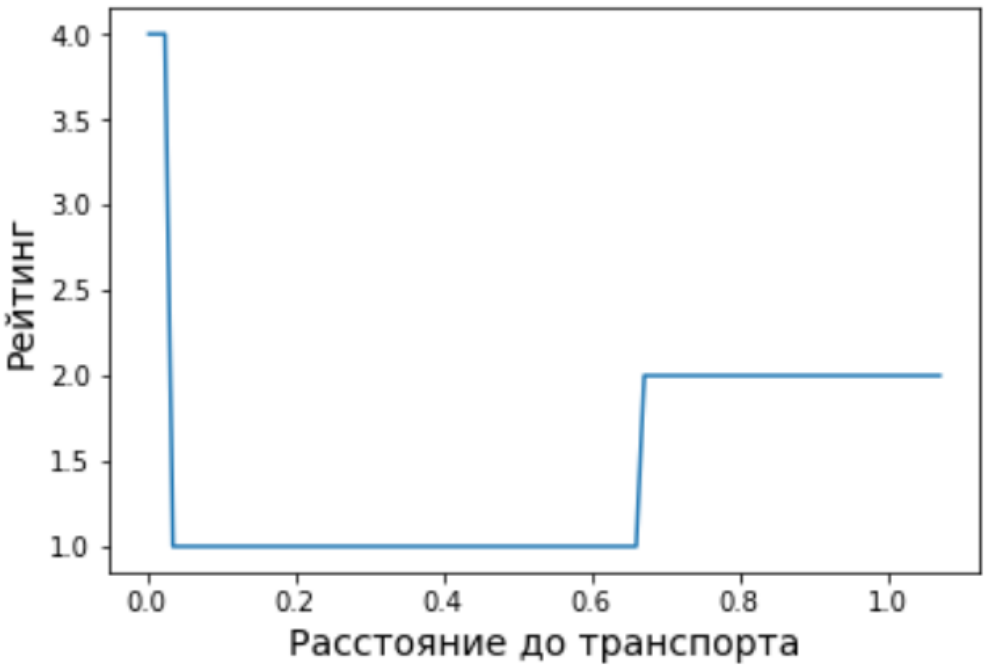
```
Test = []
Metro = np.linspace(0.01, 5.9, 100)
for i in range(len(Metro)):
    Test.append([Metro[i],t_mean,r_mean,a_mean])
Test = pd.DataFrame(Test)
# отнормируем
Test = pd.DataFrame(scal.transform(Test))
plt.plot(Metro,rfc.predict(Test))
plt.xlabel('Расстояние до метро', fontsize = 14)
plt.ylabel('Rank', fontsize = 14)
```



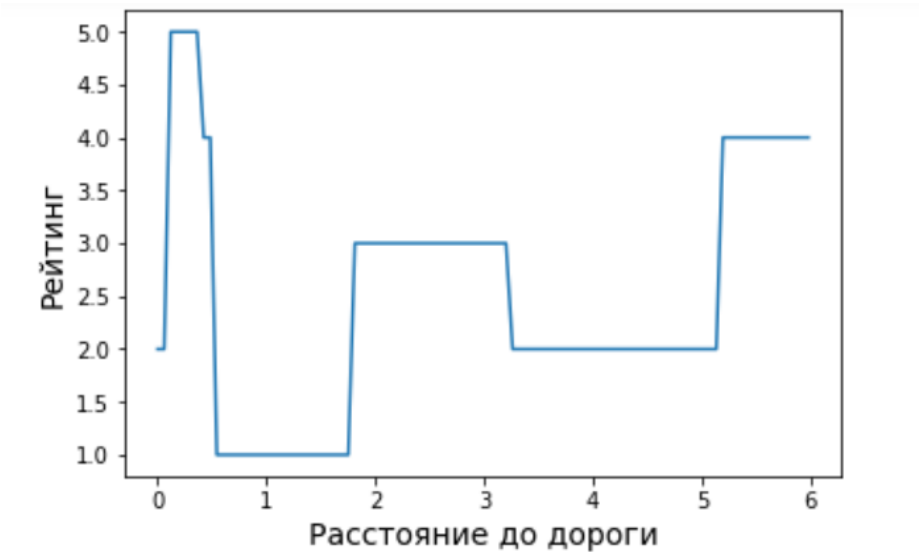
Повторим проделанную работу для трех оставшихся признаков.

```
Test = []
Transport = np.linspace(2.0e-03, 1.07, 100)
for i in range(len(Transport)):
    Test.append([m_mean,Transport[i],r_mean,a_mean])
Test = pd.DataFrame(Test)
# отнормируем
Test = pd.DataFrame(scal.transform(Test))
plt.plot(Transport,rfc.predict(Test))
```

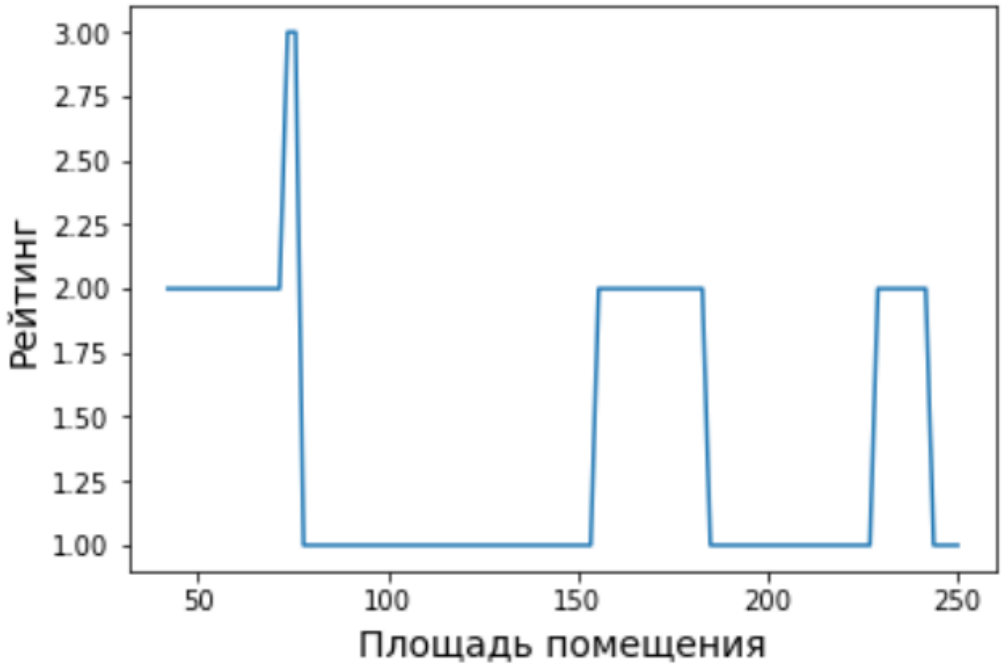
```
plt.xlabel('Расстояние до транспорта', fontsize = 14)
plt.ylabel('Рейтинг', fontsize = 14)
```



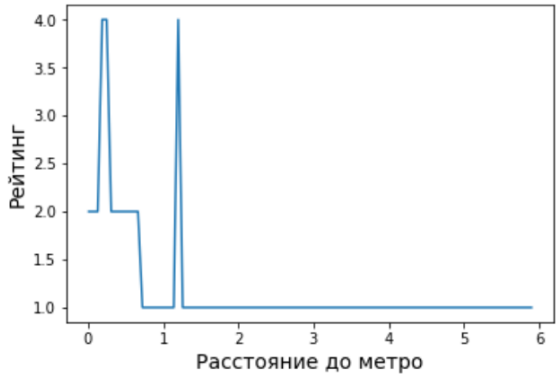

```
Test = []
Road = np.linspace(1.00000000e-03, 5.98, 100)
for i in range(len(Road)):
    Test.append([m_mean,t_mean,Road[i],a_mean])
Test = pd.DataFrame(Test)
# отнормируем
Test = pd.DataFrame(scal.transform(Test))
plt.plot(Road,rfc.predict(Test))
plt.xlabel('Расстояние до дороги', fontsize = 14)
plt.ylabel('Рейтинг', fontsize = 14)
```

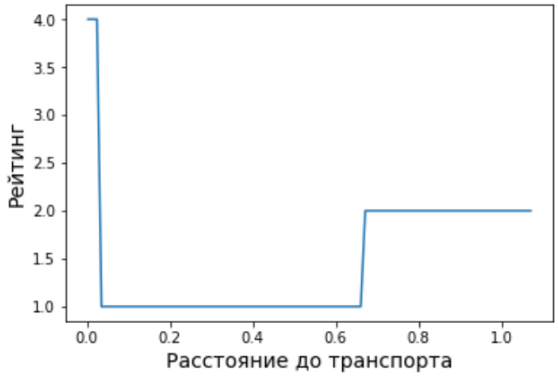


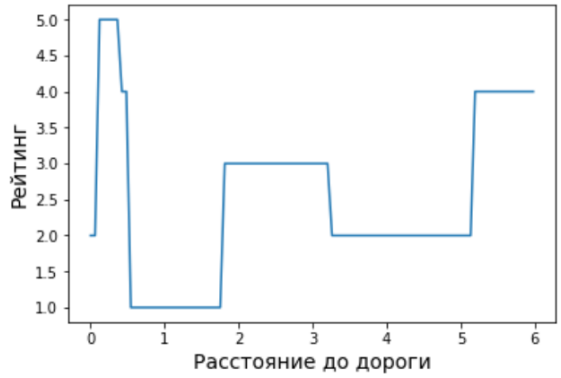

```
Test = []
Area = np.linspace(42, 250, 100)
for i in range(len(Area)):
    Test.append([m_mean,t_mean,r_mean,Area[i]])
Test = pd.DataFrame(Test)
# отнормируем
Test = pd.DataFrame(scal.transform(Test))
plt.plot(Area,rfc.predict(Test))
plt.xlabel('Площадь помещения', fontsize = 14)
plt.ylabel('Рейтинг', fontsize = 14)
```

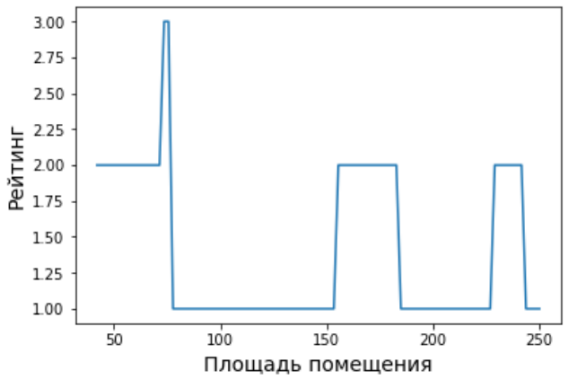


Теперь разместим графики рядом и оценим визуальной, какой из признаков доставляет больше всего рейтинга объекту.









Наиболее весомым в полученной модели оказалось расстояние до дороги, затем площадь помещение, расстояние до транспорта и наименее значимым получилось расстояние до метро. Этот результат согласуется со здравым смыслом, поскольку данные собирались не только в столице, но и в провинциальных городах, где метро развито в значительно меньшей степени, чем в Москве, поэтому и вклад у него за счет этого минимальный.

3.2 Задача регрессии

Теперь выберем оптимальную модель для задачи регрессии. Будем выбирать из двух: ***KNeighborsRegressor***[9] и ***LinearRegression***[10]**.** Заново разобьём имеющиеся данные на две выборки, причем теперь ответ берем utility.

[9] http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html
[10] http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

```
X_train, X_test, y_train, y_test
=train_test_split(df,df_ans_2,test_size=0.3,random_state=42)
```

3.2.1 KNeighborsRegressor

В этой модели реализована регрессия на основе k ближайших соседей.
Обучим модель, получим ответы для тестовой выборки, оценим эффективной модели с помощью **mean_absolute_error** [11](среднее отклонение).

```
from sklearn.neighbors import KNeighborsRegressor
neig = KNeighborsRegressor(n_neighbors=5,metric='euclidean')
neig.fit(X_train,y_train)
from sklearn.metrics import mean_absolute_error
y_pred_neig=neig.predict(X_test)
mean_absolute_error(y_test,y_pred_neig)
```

Среднее отклонение для данной модели получилось равным
`4.243679245283019`

3.2.2 LinearRegression

В модели реализован поиск линейной функции зависимости.
Обучим модель, получим ответы для тестовой выборки, оценим эффективной модели с помощью **mean_absolute_error** [12](среднее отклонение).

```
from sklearn import linear_model
linear = linear_model.LinearRegression()
linear.fit(X_train,y_train)
y_pred_linear=linear.predict(X_test)
mean_absolute_error(y_test,y_pred_linear)
```

Для данной модели среднее отклонение получилось равным
`5.0625487269814347`

[11] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_absolute_error.html
[12] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_absolute_error.html

Итак, лучшим для данной задачи оказался ***KNeighborsRegressor.***

3.2.3 Важность признаков

В этом разделе с помощью уже реализованного в scikit-learn класса ***SelectKBest***[13] получим важность признаков.

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
skb = SelectKBest(score_func=chi2, k=4)
df_ans_2=df_ans_2.astype('int')
skb.fit(df, df_ans_2)
np.set_printoptions(precision=3)
print(skb.scores_)
```

```
[  8.743   6.41   10.865   6.054]
```

Итого, самый значимый признак - удаленность от ближайшей значимой проезжей части. Далее идут удаленность от метро и удаленность от ООТ, а самый менее значимый признак - площадь отделения банка.

3.3 Выводы

В задачи классификации и в задачи регрессии получилось, что наиболее влиятельным признаком является удаленность от значимой проезжей части. При этом модель классификации получила наименее важным удаленность от метро, а модель регрессии – площадь отделения. То есть при размещении банковского отделения особое внимание стоит уделить расположению проезжих частей.
На имеющихся данных модель классификации (*RandomForestClassifier*) получилось обучить до точности в среднем 40 правильных ответов на 100 объектов. Модель регрессии (KNeighborsRegressor) дает ответ с точностью до 5 единиц.

[13] http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html

Используемые ресурсы

1) http://scikit-learn.org/stable/index.html
2) http://www.numpy.org/
3) http://pandas.pydata.org/
4) https://matplotlib.org/

Приложение 1. Данные

```
,metro,transport,road,area,utility,rank
0,0.03,0.01,0.005,75.0,24.0,4.0
1,0.88,0.38,0.62,162.0,27.0,5.0
2,1.8,0.26,0.33,85.0,28.5,5.0
3,1.1,0.18,0.25,123.0,27.7,5.0
4,0.8,0.18,0.28,94.0,29.0,5.0
5,1.3,0.2,0.1,119.0,29.1,5.0
6,0.5,0.06,0.02,97.0,26.9,5.0
7,0.3,0.02,0.006,91.0,29.1,5.0
8,0.1,0.08,0.002,63.0,21.4,3.0
9,0.44,0.33,0.89,161.0,27.5,5.0
10,4.4,0.52,1.12,104.0,12.7,1.0
11,0.36,0.65,0.22,225.0,26.2,4.0
12,0.05,0.05,0.003,88.0,26.9,5.0
13,0.1,0.15,0.15,95.0,17.5,2.0
14,0.1,0.05,0.006999999999999999,85.0,25.1,4.0
15,1.1,0.84,3.1,79.0,16.1,2.0
16,0.66,0.35,1.11,199.0,28.7,5.0
17,0.7,0.34,1.11,242.0,26.9,5.0
18,0.4,0.05,0.03,48.0,9.9,1.0
19,0.01,0.005,0.03,93.0,23.4,3.0
20,0.57,0.74,0.54,222.0,23.4,3.0
21,0.96,0.39,0.6,213.0,16.5,2.0
22,0.4,0.06,0.02,71.0,10.5,1.0
23,0.92,0.46,0.71,239.0,11.8,1.0
24,3.9,0.31,1.19,90.0,21.4,3.0
25,0.74,0.78,0.77,188.0,18.2,2.0
26,1.1,0.7,0.006,80.0,9.7,1.0
27,0.88,0.48,0.18,248.0,14.9,2.0
28,5.4,0.45,2.8,101.0,15.5,2.0
29,1.3,0.43,0.46,111.0,24.9,4.0
30,0.07,0.1,0.1,84.0,12.4,1.0
31,1.1,0.53,2.6,90.0,21.6,3.0
32,5.9,0.23,2.5,81.0,17.9,2.0
33,1.6,0.65,2.19,110.0,21.8,3.0
34,3.1,0.2,0.14,122.0,11.7,1.0
35,5.0,0.85,0.8,76.0,15.8,2.0
36,3.0,0.004,0.9,75.0,23.4,4.0
37,0.1,0.12,0.33,78.0,26.6,5.0
38,0.61,0.22,0.23,154.0,19.7,3.0
39,1.1,0.01,0.005,100.0,24.5,4.0
```

40,0.5,0.04,0.02,82.0,25.2,4.0
41,1.01,0.65,1.46,133.0,11.9,1.0
42,0.65,0.25,0.005,71.0,16.8,2.0
43,0.83,0.4,1.05,224.0,13.1,1.0
44,0.5,0.5,0.5,120.0,15.5,2.0
45,1.85,0.7,0.065,66.0,9.2,1.0
46,0.85,0.3,0.08,87.0,19.35,3.0
47,0.65,0.22,1.14,248.0,21.8,3.0
48,0.55,0.37,0.45,103.0,12.4,1.0
49,0.68,0.7,1.25,218.0,20.2,3.0
50,0.85,0.06,0.06,90.0,23.7,4.0
51,2.7,0.58,0.09,74.0,11.0,1.0
52,0.85,0.4,0.2,80.0,11.8,1.0
53,0.35,0.07,0.002,103.0,28.7,5.0
54,0.8,0.05,0.001,65.0,15.7,2.0
55,0.72,1.07,0.84,192.0,12.2,1.0
56,2.1,0.14,0.92,96.0,11.6,1.0
57,1.0,0.3,0.3,80.0,15.2,2.0
58,0.4,0.2,0.1,65.0,20.5,3.0
59,0.09,0.4,0.84,208.0,24.8,4.0
60,0.85,0.43,0.18,191.0,29.2,5.0
61,0.1,0.12,0.65,71.0,11.7,1.0
62,0.9,0.24,0.12,119.0,11.4,1.0
63,0.8,0.54,1.07,234.0,26.8,5.0
64,0.2,0.15,0.15,68.0,12.4,1.0
65,0.6,0.45,0.035,82.0,17.0,2.0
66,0.7,0.7,0.005,80.0,11.1,1.0
67,0.6,0.2,0.2,84.0,12.3,1.0
68,0.05,0.02,0.001,90.0,28.2,5.0
69,0.76,0.47,0.51,118.0,13.2,1.0
70,0.5,0.3,0.004,42.0,8.5,1.0
71,0.45,0.36,0.2,110.0,15.2,2.0
72,0.74,0.78,0.49,165.0,22.9,3.0
73,0.6,0.06,0.06,78.0,15.7,2.0
74,0.65,0.25,0.3,80.0,16.1,2.0
75,0.06,0.01,0.004,98.0,29.4,5.0
76,3.5,0.79,5.54,107.0,20.5,3.0
77,4.3,0.47,0.44,98.0,11.7,1.0
78,0.8,0.33,0.77,211.0,10.2,1.0
79,0.86,0.45,1.3,124.0,12.0,1.0
80,0.05,0.39,0.85,199.0,23.8,4.0
81,2.5,0.24,2.82,71.0,20.7,3.0
82,0.08,0.02,0.003,83.0,29.5,5.0
83,0.05,0.15,0.05,67.0,13.1,1.0
84,0.9,0.15,0.06,88.0,18.7,2.0
85,3.7,0.69,1.14,121.0,10.9,1.0
86,0.65,0.07,0.04,63.0,10.1,1.0
87,1.4,0.25,0.005,51.0,10.4,1.0
88,1.5,0.11,3.4,82.0,19.2,2.0
89,0.74,0.28,1.46,250.0,11.8,1.0
90,0.04,0.03,0.002,95.0,29.1,5.0
91,2.3,0.45,0.75,95.0,10.8,1.0
92,0.58,0.41,0.9,174.0,16.3,2.0
93,1.1,0.07,0.002,98.0,22.5,3.0
94,0.53,0.27,0.81,193.0,16.5,2.0
95,0.96,0.35,0.49,184.0,28.2,5.0

96,3.4,0.81,4.46,107.0,26.4,4.0
97,0.05,0.02,0.002,86.0,28.3,5.0
98,2.0,0.2,0.11,93.0,27.7,5.0
99,1.1,0.24,5.82,123.0,26.1,4.0
100,0.39,0.53,0.96,225.0,24.3,4.0
101,0.04,0.025,0.01,105.0,27.1,5.0
102,0.83,0.37,0.54,195.0,21.7,3.0
103,0.32,0.34,0.08,193.0,27.5,5.0
104,2.5,0.64,0.34,101.0,20.3,3.0
105,0.85,0.38,0.78,157.0,17.7,2.0
106,1.02,0.34,0.89,198.0,12.5,1.0
107,0.95,0.04,0.002,82.0,25.8,4.0
108,2.4,0.2,0.42,79.0,13.6,2.0
109,0.24,0.5,0.5,101.0,15.8,2.0
110,0.4,0.16,0.29,79.0,27.2,5.0
111,0.16,0.2,0.35,63.0,11.1,1.0
112,0.36,0.39,0.08,245.0,24.1,4.0
113,1.1,0.35,0.05,85.0,17.5,2.0
114,3.6,0.3,5.7,106.0,13.6,2.0
115,1.0,0.01,0.002,65.0,20.3,3.0
116,0.54,0.8,0.61,222.0,14.3,2.0
117,0.72,0.79,0.75,153.0,22.4,3.0
118,0.43,0.17,0.14,82.0,13.5,1.0
119,0.6,0.1,0.2,75.0,20.1,3.0
120,1.2,0.94,0.97,111.0,10.0,1.0
121,0.04,0.01,0.003,93.0,28.9,5.0
122,5.6,0.29,0.51,128.0,10.2,1.0
123,3.3,0.75,5.32,103.0,24.1,4.0
124,3.2,0.94,0.88,82.0,20.0,3.0
125,0.25,0.49,0.5,179.0,26.4,4.0
126,1.9,0.41,3.01,99.0,22.7,3.0
127,1.2,0.18,4.54,82.0,25.7,4.0
128,5.5,0.91,0.5,70.0,14.1,2.0
129,0.15,0.23,0.4,85.0,16.1,2.0
130,1.2,0.53,5.7,118.0,15.0,2.0
131,0.9,0.11,0.001,80.0,21.2,3.0
132,0.3,0.3,0.15,90.0,21.8,3.0
133,0.7,0.3,0.3,100.0,13.9,2.0
134,1.6,0.81,2.54,112.0,21.7,3.0
135,0.27,0.69,0.68,247.0,25.2,4.0
136,0.75,0.19,1.11,134.0,12.0,1.0
137,0.47,0.6,0.69,235.0,27.5,5.0
138,0.2,0.28,0.11,101.0,27.6,5.0
139,1.6,0.056,0.01,102.0,15.6,3.0
140,0.56,0.53,1.17,215.0,22.5,3.0
141,0.8,0.25,0.02,105.0,19.1,2.0
142,0.9,0.45,0.45,55.0,13.5,2.0
143,1.8,0.71,3.58,88.0,24.5,4.0
144,1.9,0.23,0.22,106.0,27.7,5.0
145,0.61,0.27,1.18,179.0,14.0,2.0
146,0.1,0.35,0.001,85.0,26.8,5.0
147,0.8,0.01,0.004,69.0,23.6,4.0
148,1.05,0.2,0.005,70.0,9.2,1.0
149,0.3,0.3,0.3,73.0,11.7,1.0
150,1.55,0.25,0.015,56.0,7.3,1.0
151,0.24,0.65,0.66,219.0,24.7,4.0

152,0.1,0.1,0.1,80.0,22.2,3.0
153,1.5,0.1,0.2,88.0,22.4,3.0
154,0.62,0.35,0.05,163.0,19.2,2.0
155,0.17,0.48,0.1,227.0,26.4,4.0
156,0.3,0.05,0.05,85.0,17.0,2.0
157,3.0,0.21,0.4,110.0,12.5,1.0
158,2.6,0.31,2.64,92.0,26.2,4.0
159,0.92,0.72,0.95,151.0,18.9,2.0
160,0.6,0.05,0.05,85.0,19.1,2.0
161,0.5,0.2,0.2,65.0,15.4,2.0
162,0.25,0.25,0.25,45.0,14.0,2.0
163,0.62,0.4,1.3,183.0,23.3,3.0
164,0.4,0.7,0.9,91.0,12.8,1.0
165,0.9,0.18,0.34,91.0,27.5,5.0
166,0.45,0.1,0.015,78.0,13.2,1.0
167,0.2,0.4,0.4,105.0,21.0,3.0
168,0.07,0.05,0.001,79.0,23.3,3.0
169,0.62,0.56,1.19,216.0,29.8,5.0
170,0.45,0.34,1.06,186.0,29.0,5.0
171,3.1,0.64,1.1,77.0,10.8,1.0
172,0.4,0.015,0.01,62.0,11.9,1.0
173,0.66,0.71,0.86,244.0,20.4,3.0
174,0.85,0.02,0.006999999999999999,86.0,24.3,4.0
175,1.2,0.12,5.62,73.0,26.4,4.0
176,0.59,0.27,0.51,241.0,17.5,2.0
177,0.64,0.52,0.03,237.0,19.9,3.0
178,0.8,0.49,0.28,135.0,13.3,1.0
179,0.6,0.06,0.02,81.0,26.2,4.0
180,0.63,0.31,1.18,245.0,21.4,3.0
181,0.15,0.2,0.15,95.0,13.3,1.0
182,0.71,0.67,0.1,178.0,28.8,5.0
183,0.9,0.065,0.05,93.0,20.1,3.0
184,0.04,0.02,0.01,85.0,19.5,3.0
185,0.59,0.24,0.42,156.0,19.7,3.0
186,0.08,0.46,0.86,209.0,26.1,4.0
187,4.6,0.44,0.61,90.0,13.3,1.0
188,1.05,0.2,0.04,65.0,14.1,2.0
189,0.07,0.02,0.001,84.0,26.3,4.0
190,1.18,0.77,1.26,214.0,12.6,1.0
191,0.85,0.45,0.05,55.0,8.1,1.0
192,2.7,0.33,0.1,82.0,12.6,1.0
193,1.6,0.13,0.66,99.0,22.2,3.0
194,0.3,0.6,0.05,60.0,13.6,2.0
195,0.6,0.5,2.56,109.0,24.2,4.0
196,0.09,0.05,0.004,75.0,27.3,5.0
197,0.3,0.2,0.15,69.0,12.3,1.0
198,2.1,0.4,0.001,50.0,18.8,2.0
199,1.3,0.005,0.001,70.0,16.3,2.0
200,0.8,0.27,5.76,105.0,21.1,3.0
201,0.7,0.3,0.02,67.0,19.4,2.0
202,0.7,0.6,0.01,70.0,12.1,1.0
203,0.025,0.002,0.001,99.0,29.4,5.0
204,0.2,0.05,0.015,98.0,24.4,4.0
205,0.07,0.075,0.03,98.0,23.3,3.0
206,0.1,0.3,0.3,85.0,22.5,3.0
207,0.05,0.05,0.1,95.0,19.0,2.0

208,2.4,0.51,0.45,122.0,23.5,4.0
209,1.2,0.69,0.6,99.0,14.1,2.0
210,0.7,0.4,0.01,60.0,11.7,1.0
211,0.33,0.45,0.69,197.0,27.8,5.0
212,0.55,0.65,0.05,110.0,17.1,2.0
213,0.7,0.67,0.66,168.0,22.4,3.0
214,0.8,0.6,0.06,97.0,17.1,2.0
215,0.7,0.05,0.02,91.0,24.8,4.0
216,0.84,0.4,1.33,221.0,13.4,1.0
217,1.2,0.65,0.76,216.0,12.7,1.0
218,4.1,0.41,0.27,94.0,11.4,1.0
219,4.1,0.82,1.6,70.0,16.6,2.0
220,0.9,0.35,0.48,206.0,14.4,2.0
221,0.3,0.2,0.006,90.0,25.4,4.0
222,0.8,0.1,0.1,118.0,27.4,5.0
223,3.4,0.16,3.3,109.0,14.5,2.0
224,0.52,0.32,1.3,216.0,17.6,2.0
225,0.2,0.03,0.004,84.0,26.1,4.0
226,0.55,0.4,0.15,56.0,12.3,1.0
227,0.8,0.1,0.19,105.0,27.5,5.0
228,1.2,0.79,1.26,107.0,22.2,3.0
229,0.13,0.015,0.05,100.0,13.4,1.0
230,1.4,0.65,0.85,125.0,19.8,3.0
231,0.64,0.76,1.23,242.0,18.5,2.0
232,0.55,0.05,0.005,92.0,25.2,4.0
233,1.2,0.05,0.005,85.0,14.5,2.0
234,0.95,0.55,0.005,52.0,8.1,1.0
235,1.0,0.86,0.25,184.0,10.4,1.0
236,1.2,0.49,0.69,188.0,11.8,1.0
237,0.35,0.1,0.1,60.0,23.1,3.0
238,0.6,0.03,0.03,77.0,25.7,4.0
239,1.4,0.29,0.2,112.0,29.1,5.0
240,0.85,0.01,0.25,48.0,18.2,2.0
241,0.69,0.62,1.24,217.0,15.9,2.0
242,0.6,0.3,0.03,45.0,7.8,1.0
243,0.7,0.49,0.72,86.0,22.3,3.0
244,3.5,0.29,3.4,121.0,19.1,2.0
245,3.7,0.62,4.43,77.0,23.5,4.0
246,0.15,0.1,0.1,90.0,18.0,2.0
247,0.52,0.51,0.55,160.0,26.9,5.0
248,0.9,0.44,1.04,158.0,13.9,2.0
249,0.82,0.52,0.32,247.0,22.2,3.0
250,0.05,0.01,0.005,86.0,25.7,4.0
251,3.0,0.83,0.67,101.0,12.4,1.0
252,2.2,0.94,1.72,107.0,20.1,3.0
253,0.2,0.3,0.18,85.0,28.4,5.0
254,0.49,0.64,1.25,216.0,17.4,2.0
255,0.8,0.4,0.35,70.0,15.5,2.0
256,0.96,0.44,0.61,239.0,13.8,2.0
257,0.5,0.1,0.3,82.0,11.9,1.0
258,2.3,0.53,4.96,126.0,26.2,4.0
259,0.5,0.43,2.6,90.0,14.7,2.0
260,0.35,0.01,0.01,75.0,19.0,2.0
261,0.52,0.49,0.33,160.0,16.4,2.0
262,2.2,0.8,0.035,69.0,13.5,2.0
263,0.85,0.4,0.015,63.0,15.2,2.0

264,2.9,0.8,1.2,86.0,20.8,3.0
265,0.7,0.35,0.35,76.0,22.3,3.0
266,0.35,0.15,0.15,70.0,21.5,3.0
267,0.26,0.44,0.7,231.0,23.6,4.0
268,2.3,0.41,3.24,101.0,24.0,4.0
269,0.48,0.54,0.95,87.0,11.5,1.0
270,0.1,0.53,0.4,185.0,25.7,4.0
271,0.4,0.35,0.001,77.0,8.85,1.0
272,0.3,0.62,0.07,191.0,23.5,4.0
273,0.3,0.36,0.36,95.0,17.5,2.0
274,0.5,0.39,5.63,116.0,21.8,3.0
275,0.3,0.2,0.05,85.0,25.2,4.0
276,0.82,0.42,1.17,175.0,28.1,5.0
277,0.1,0.18,0.14,97.0,26.6,5.0
278,0.41,0.64,0.72,157.0,18.3,2.0
279,0.52,0.63,0.85,155.0,29.7,5.0
280,0.55,0.25,0.25,63.0,16.2,2.0
281,0.79,0.65,0.74,242.0,26.4,5.0
282,0.4,0.68,1.15,204.0,28.1,5.0
283,0.65,0.6,0.01,62.0,12.2,1.0
284,0.85,0.7,0.001,90.0,18.7,2.0
285,1.2,0.06,0.01,69.0,20.2,3.0
286,3.5,0.3,0.5,101.0,12.1,1.0
287,0.1,0.48,0.4,168.0,24.2,4.0
288,0.8,0.22,0.14,117.0,27.0,5.0
289,0.75,0.1,0.001,60.0,9.4,1.0
290,0.7,0.54,2.1,107.0,23.9,4.0
291,0.99,0.46,0.33,167.0,27.9,5.0
292,0.58,0.36,0.36,90.0,16.7,2.0
293,0.25,0.42,1.13,197.0,25.3,4.0
294,0.69,0.62,0.94,197.0,11.6,1.0
295,2.3,0.67,4.21,100.0,24.9,4.0
296,2.3,0.38,4.46,118.0,22.9,3.0
297,0.08,0.05,0.03,100.0,23.2,3.0
298,0.1,0.29,0.36,70.0,27.2,5.0
299,4.0,0.67,3.6,96.0,26.1,4.0
300,0.035,0.03,0.015,62.0,22.35,3.0
301,0.3,0.68,1.12,230.0,26.4,5.0
302,0.3,0.03,0.02,88.0,23.1,3.0
303,0.55,0.35,0.35,55.0,16.3,2.0
304,3.9,0.9,0.21,122.0,11.0,1.0
305,0.57,0.29,0.75,86.0,12.3,1.0
306,4.3,0.92,4.5,76.0,15.2,2.0
307,3.1,0.55,0.15,70.0,10.1,1.0
308,0.67,0.65,1.05,243.0,21.7,3.0
309,0.76,0.3,0.13,77.0,12.7,1.0
310,0.25,0.55,0.15,169.0,24.0,4.0
311,1.2,0.7,0.03,83.0,16.2,2.0
312,0.8,0.4,0.001,74.0,8.8,1.0
313,1.7,0.26,0.16,125.0,29.9,5.0
314,0.65,0.04,0.006999999999999999,100.0,19.3,2.0
315,0.76,0.67,0.79,113.0,11.7,1.0
316,4.5,0.65,0.19,71.0,11.6,1.0
317,2.7,0.54,0.39,123.0,11.0,1.0
318,4.8,0.87,1.33,75.0,11.5,1.0
319,0.36,0.52,0.43,164.0,25.1,4.0

320,2.0,0.92,5.98,80.0,23.8,4.0
321,1.2,0.79,5.0,109.0,18.9,2.0
322,0.7,0.2,0.07,82.0,12.5,1.0
323,0.15,0.31,0.79,190.0,24.5,4.0
324,0.38,0.51,0.18,247.0,24.4,4.0
325,0.67,0.67,0.91,240.0,21.1,3.0
326,0.3,0.03,0.005,100.0,27.5,5.0
327,0.62,0.89,1.25,160.0,22.3,1.0
328,0.1,0.03,0.01,65.0,29.1,5.0
329,1.3,0.4,0.15,45.0,10.8,1.0
330,0.04,0.009000000000000001,0.002,77.0,28.6,5.0
331,0.55,0.71,0.22,206.0,20.5,3.0
332,0.4,0.07,0.01,91.0,23.8,4.0
333,5.5,0.39,3.5,117.0,17.0,2.0
334,3.9,0.77,1.88,77.0,20.3,3.0
335,3.0,0.18,5.98,77.0,25.8,4.0
336,0.7,0.04,0.04,90.0,21.3,3.0
337,0.11,0.61,0.45,198.0,23.5,4.0
338,4.1,0.83,3.5,128.0,16.6,2.0
339,0.1,0.2,0.05,65.0,12.1,1.0
340,0.4,0.01,0.001,80.0,22.9,3.0
341,0.3,0.1,0.8,90.0,10.3,2.0
342,0.3,0.25,0.25,90.0,12.6,1.0
343,0.73,0.22,1.07,214.0,23.0,3.0
344,2.1,0.49,0.8,118.0,15.7,2.0
345,0.07,0.03,0.004,95.0,29.6,5.0
346,0.7,0.4,0.006,94.0,23.7,4.0
347,1.4,0.08,0.001,73.0,19.7,3.0
348,0.5,0.47,5.68,88.0,23.9,4.0
349,0.8,0.04,0.005,92.0,27.8,5.0
350,0.56,0.74,0.5,239.0,23.4,3.0
351,0.05,0.05,0.015,77.0,21.8,3.0
352,1.3,0.35,0.005,82.0,20.8,3.0