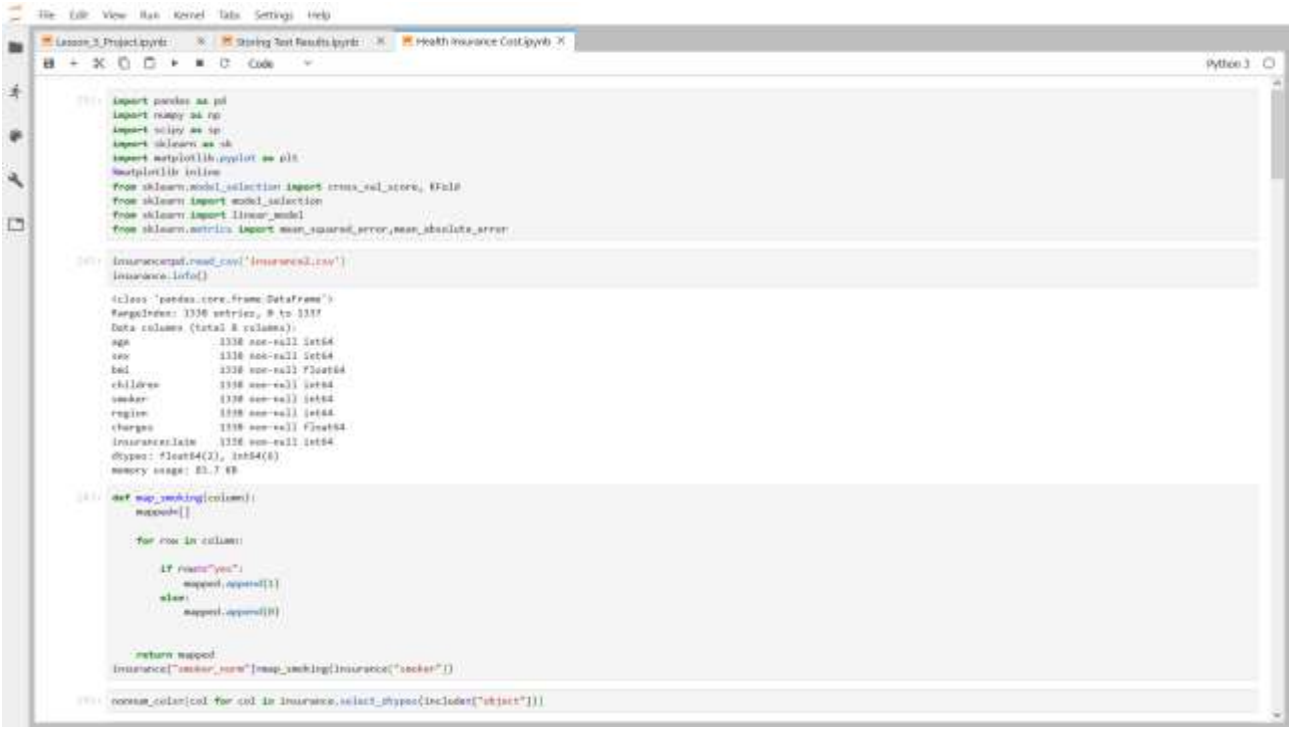
SCREENSHOTS



```
import pandas as pd
import numpy as np
import scipy as sp
import sklearn as sk
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import cross_val_score, KFold
from sklearn import model_selection
from sklearn import linear_model
from sklearn.metrics import mean_squared_error,mean_absolute_error
```

```
insurance=pd.read_csv('insurance2.csv')
insurance.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 8 columns):
age               1338 non-null int64
sex               1338 non-null int64
bmi               1338 non-null float64
children          1338 non-null int64
smoker            1338 non-null int64
region            1338 non-null int64
charges           1338 non-null float64
insuranceclaim    1338 non-null int64
dtypes: float64(2), int64(6)
memory usage: 83.7 KB
```

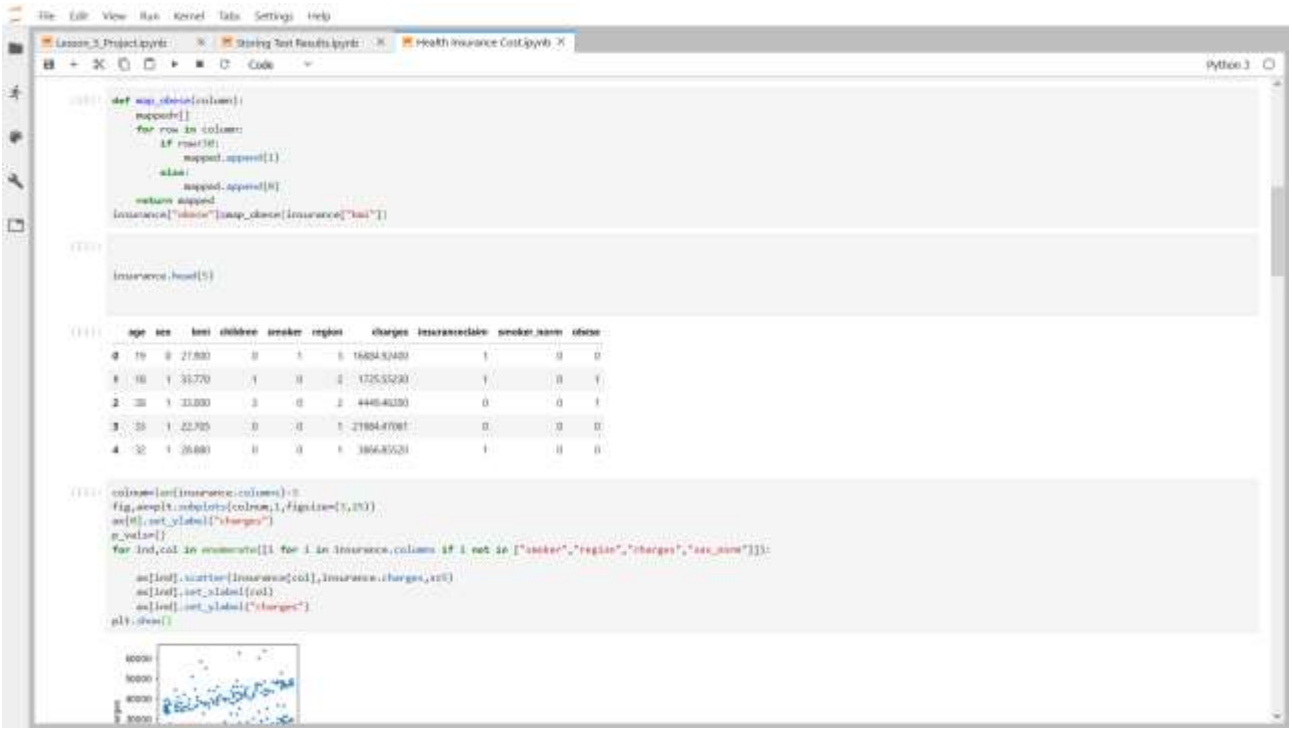
```
def map_smoking(column):
    mapped=[]

    for row in column:

        if row=="yes":
            mapped.append(1)
        else:
            mapped.append(0)


    return mapped
insurance["smoker_norm"]=map_smoking(insurance["smoker"])
```

```
numcols=[col for col in insurance.select_dtypes(include=["object"])]
```



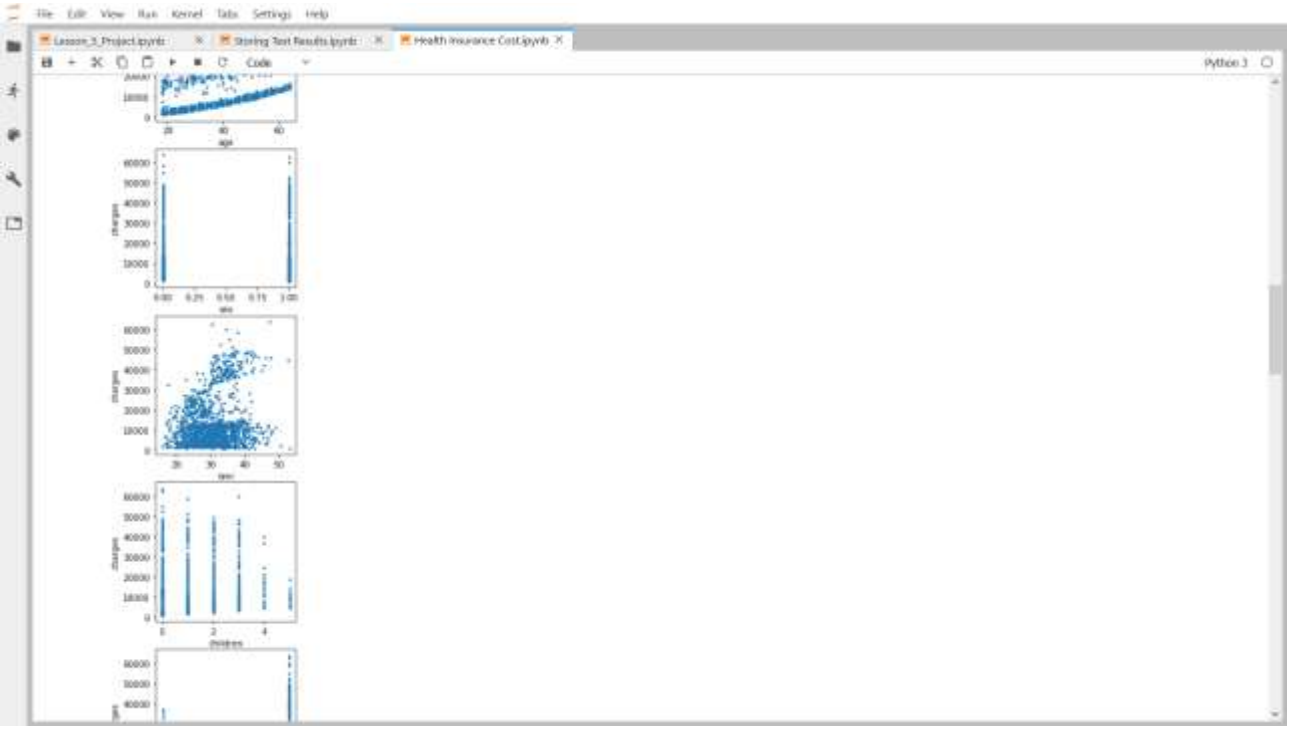
```
def map_obese(column):
    mapped=[]
    for row in column:
        if row>30:
            mapped.append(1)
        else:
            mapped.append(0)
    return mapped
insurance["obese"]=map_obese(insurance["bmi"])
```

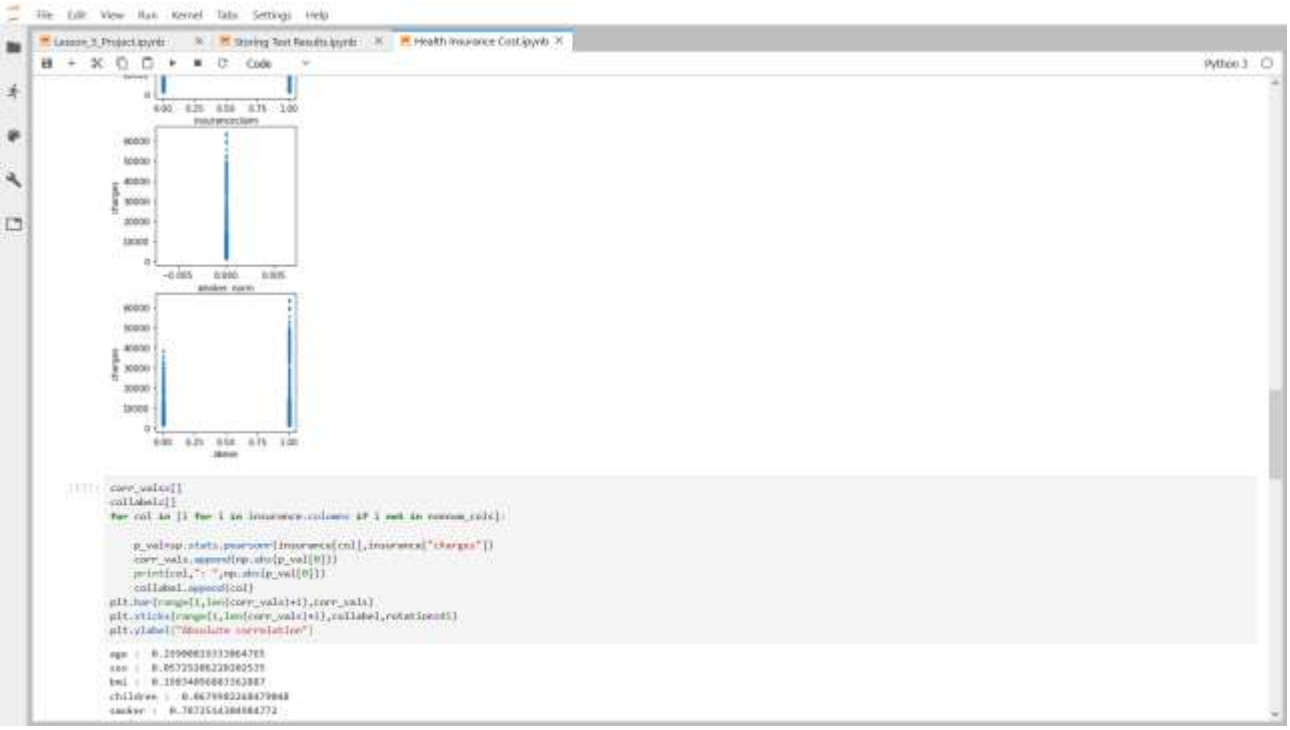
```
insurance.head(5)
```

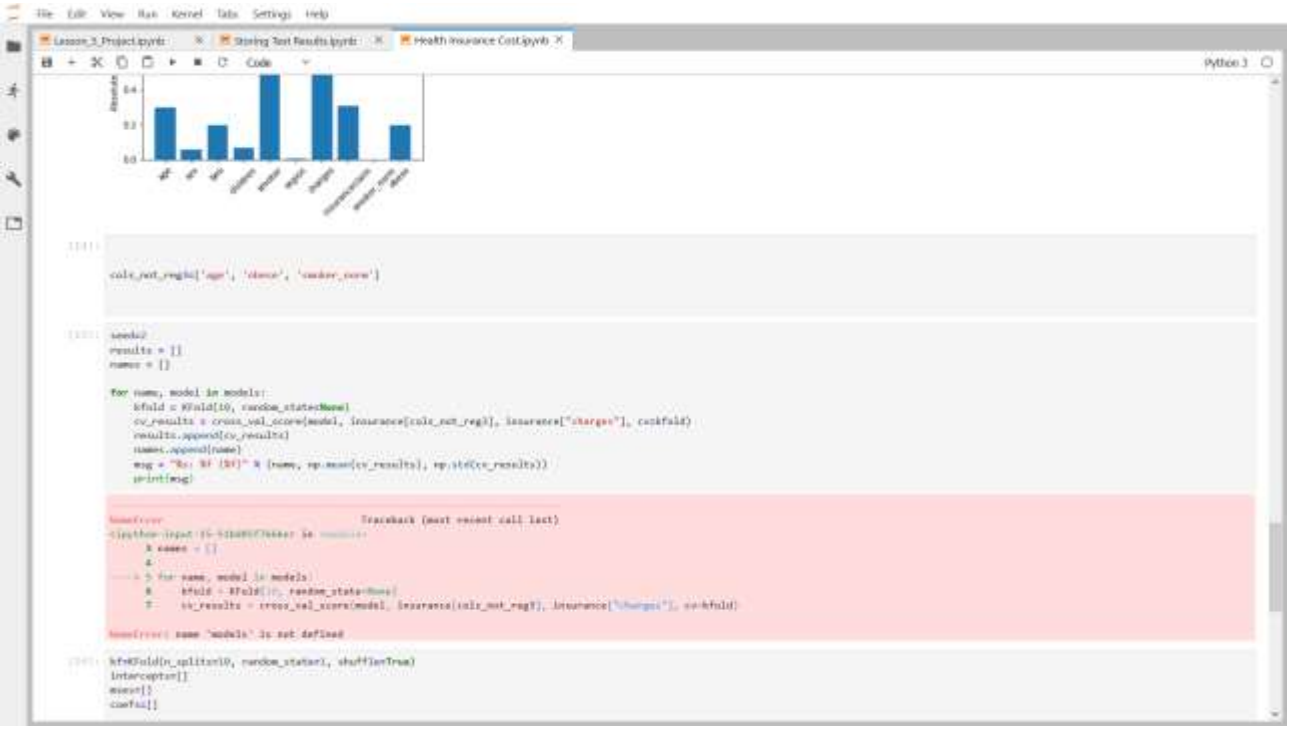
	age	sex	bmi	children	smoker	region	charges	insuranceclaim	smoker_norm	obese
0	19	0	27.900	0	1	3	16884.92400	1	0	0
1	18	1	33.770	1	0	2	1725.55230	1	0	1
2	28	1	33.000	3	0	2	4449.46200	0	0	1
3	33	1	22.705	0	0	1	21984.47061	0	0	0
4	32	1	28.880	0	0	1	3866.85520	1	0	0

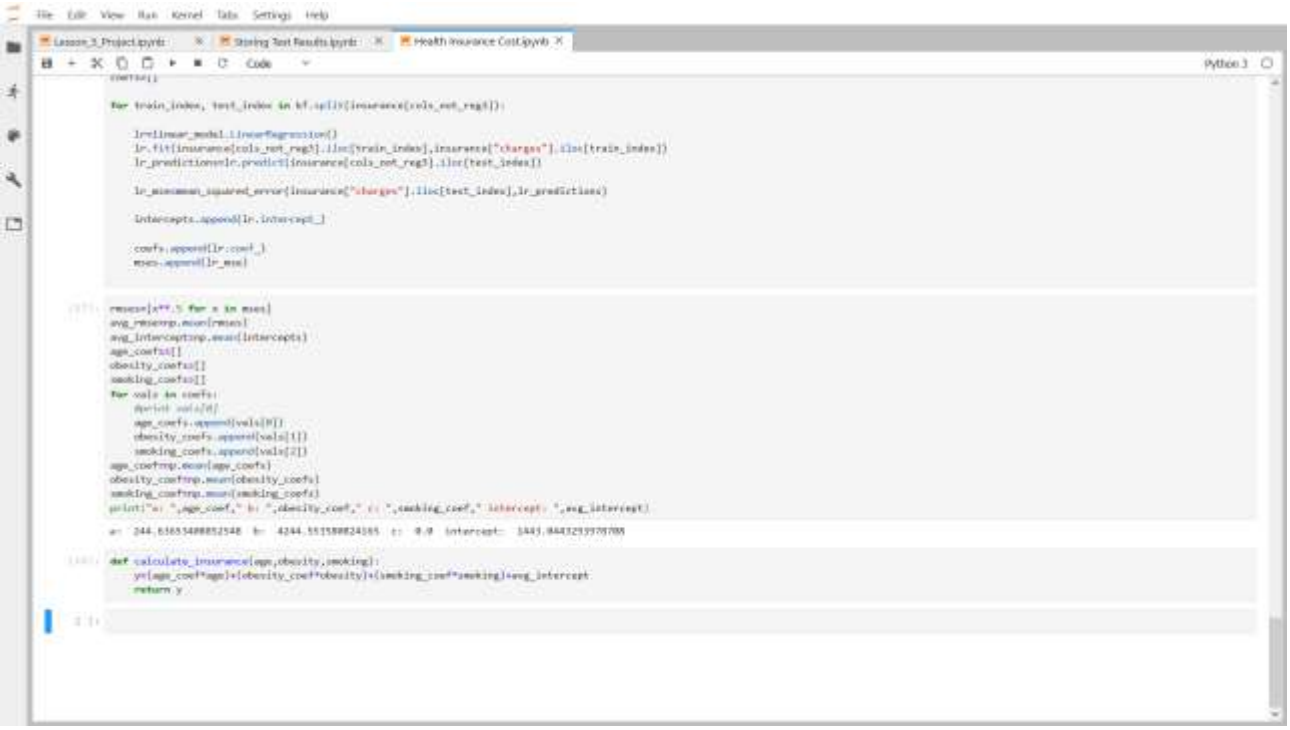
```
colnum=len(insurance.columns)-3
fig,axsplt.subplots(colnum,1,figsize=(5,25))
ax[0].set_ylabel("charges")
p_value=[]
for ind,col in enumerate([i for i in insurance.columns if i not in ["smoker","region","charges","sex_norm"]]):

    ax[ind].scatter(insurance[col],insurance.charges,s=5)
    ax[ind].set_xlabel(col)
    ax[ind].set_ylabel("charges")
plt.show()
```


File Edit View Run Kernel Tabs Settings Help
Lesson_3_Project.ipynb
Storing Test Results.ipynb
Health Insurance Cost.ipynb
Code
Python 3
age
sex
bmi
children


File Edit View Run Kernel Tabs Settings Help
Lesson_3_Project.ipynb
Storing Test Results.ipynb
Health Insurance Cost.ipynb
Code
Python 3
insurancecharges
smoker_num
charges
corr_vals=[]
collabels=[]
for col in [i for i in insurance.columns if i not in numcols]:
p_val=sp.stats.pearsonr(insurance[col],insurance["charges"])
corr_vals.append(np.abs(p_val[0]))
print(col,": ",np.abs(p_val[0]))
collabel.append(col)
plt.bar(range(1,len(corr_vals)+1),corr_vals)
plt.xticks(range(1,len(corr_vals)+1),collabel,rotation=45)
plt.ylabel("Absolute correlation")
age : 0.299008193166765
sex : 0.057293062208382579
bmi : 0.198340968675363887
children : 0.067998223648799848
smoker : 0.787251430498477272





SOURCE CODE

```
import pandas as pd
import numpy as np
import scipy as sp
import sklearn as sk
```

```
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import cross_val_score, KFold
from sklearn import model_selection
from sklearn import linear_model
from sklearn.metrics import mean_squared_error,mean_absolute_error
```

Loading the data

```
insurance=pd.read_csv("insurance2.csv")
insurance.info()
def map_smoking(column):
    mapped=[]

    for row in column:

        if row=="yes":
            mapped.append(1)
        else:
            mapped.append(0)


    return mapped
insurance["smoker_norm"]=map_smoking(insurance["smoker"])
nonnum_cols=[col for col in insurance.select_dtypes(include=["object"])]
def map_obese(column):
    mapped=[]
    for row in column:
        if row>30:
            mapped.append(1)
        else:
            mapped.append(0)
    return mapped
insurance["obese"]=map_obese(insurance["bmi"])
insurance.head(5)
```

We now explore the relation between the features given and the insurance costs

```
colnum=len(insurance.columns)-3
fig,ax=plt.subplots(colnum,1,figsize=(3,25))
ax[0].set_ylabel("charges")
p_vals={}
for ind,col in enumerate([i for i in insurance.columns if i not in
["smoker","region","charges","sex_norm"]]):

    ax[ind].scatter(insurance[col],insurance.charges,s=5)
    ax[ind].set_xlabel(col)
    ax[ind].set_ylabel("charges")
plt.show()

corr_vals=[]
collabel=[]
for col in [i for i in insurance.columns if i not in nonnum_cols]:
```

```
    p_val=sp.stats.pearsonr(insurance[col],insurance["charges"])
    corr_vals.append(np.abs(p_val[0]))
    print(col,": ",np.abs(p_val[0]))
    collabel.append(col)
plt.bar(range(1,len(corr_vals)+1),corr_vals)
plt.xticks(range(1,len(corr_vals)+1),collabel,rotation=45)
plt.ylabel("Absolute correlation")

cols_not_reg3=['age', 'obese', 'smoker_norm']
seed=2
results = []
names = []

for name, model in models:
    kfold = KFold(10, random_state=None)
    cv_results = cross_val_score(model, insurance[cols_not_reg3],
insurance["charges"], cv=kfold)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, np.mean(cv_results), np.std(cv_results))
    print(msg)

kf=KFold(n_splits=10, random_state=1, shuffle=True)
intercepts=[]
mses=[]
coefs=[]

for train_index, test_index in kf.split(insurance[cols_not_reg3]):

    lr=linear_model.LinearRegression()

lr.fit(insurance[cols_not_reg3].iloc[train_index],insurance["charges"].iloc[trai
n_index])
    lr_predictions=lr.predict(insurance[cols_not_reg3].iloc[test_index])


lr_mse=mean_squared_error(insurance["charges"].iloc[test_index],lr_predictions)

    intercepts.append(lr.intercept_)

    coefs.append(lr.coef_)
    mses.append(lr_mse)

rmses=[x**.5 for x in mses]
avg_rmse=np.mean(rmses)
avg_intercept=np.mean(intercepts)
age_coefs=[]
obesity_coefs=[]
smoking_coefs=[]
for vals in coefs:
    #print vals[0]
    age_coefs.append(vals[0])
    obesity_coefs.append(vals[1])
    smoking_coefs.append(vals[2])
age_coef=np.mean(age_coefs)
```

```
obesity_coef=np.mean(obesity_coefs)
smoking_coef=np.mean(smoking_coefs)
print("a: ",age_coef," b: ",obesity_coef," c: ",smoking_coef," intercept:
",avg_intercept)
```