BIG DATA ECONOMETRIC FINAL EXAM PROJECT

DEEPAK KUMAR TIWARI

ECG-590

NORTH CAROLINA STATE UNIVERSITY

FINANCIAL MATHEMATICS, DEC'19

Goal of this project is to reduce the Mean squared Error using different Machine Learning and Deep Learning methods

Takeaway from this project:

1. Quadratic Model fitted best on this data with MSE of 18%
2. Random Forest performed better than than the Decision Tree or Linear Regression with all 50 features because of vote of several independent trees in the forest
3. Train Test split is a good method to measure the model accuracy on unseen data but since here out data is small and has only 400 data points, 50% ratio is not good enough we would have as much data used to train the model. only 200 datapoints to train the model is not the best idea
4. There is multicollinearity in the variables but if we lose the variables expecially first 29 variables we are losing alot of information thus MSE increased from 23(Linear Regression when all 50 variables used) to 28(When last 21 variables used). VIF above 10 indicates multicollinearity
5. Ridge Regression with penalty parameter of 1 worked better than the Linear Regression as well as Lasso Regression
6. Lasso Regression is very important method to select the variables as it converges the parameter cofficient to zero at different penalty parameter lambda
7. Lasso Regression chose 20 variables out of 50 variables and MSE also improved from 23% when all 50 variables were used to 20% when only 20 variables were used in the model
8. Artificial Neural Network model redcuced the MSE to 7.1% on training dataset with only 200 points but performing not that good on test dataset. More training dataset might improve the model performance. By cross validation we can train the model on datapoints and also test the model on data but we are not allowed to do that in this exam project.
9. Best subset can be a good algorithm to find best models but when we have too many idependent variables like here we have 50 variables , it becomes computationally unfeasible for 3 predictors only we have 19000 models. We can alternatively use forward, backward or mixed stepwise selection.

Let's start by importing libraries

In [0]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.model_selection import train_test_split
```

Let's get the data file

In [4]:
```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Data.csv to Data.csv
```

In [0]:
```
import io
ExamData = pd.read_csv(io.BytesIO(uploaded['Data.csv']))
```

Let's check the data once

In [6]:
```
ExamData.head()
```

Out[6]:

	y	x01	x02	x03	x04	x05	x06	x07	x08	x09	x10	
0	-3.987019	-1.150161	1.919942	-0.200096	-0.144521	-0.482146	-0.960322	-1.817256	-0.650505	-1.263704	-0.504251	-1.(
1	4.385074	0.202677	1.184966	0.554749	1.440169	0.775009	0.645686	-0.504184	0.410975	-0.027140	-0.512595	0.1
2	8.190520	0.821439	1.066287	0.380227	0.115380	-0.057053	-1.254392	0.431652	1.063195	-2.030637	-0.220205	0.9
3	0.718460	1.051590	-0.854382	0.460841	-1.645333	2.325046	0.840424	0.032657	0.082418	-0.401942	-0.247423	1.0
4	-7.689298	-1.017523	-0.592204	-0.896561	1.946834	-0.740624	-0.814975	-0.011092	-0.593935	-2.546734	-1.984688	-1.(

In [7]:
```
ExamData.shape
```

Out[7]: (400, 51)

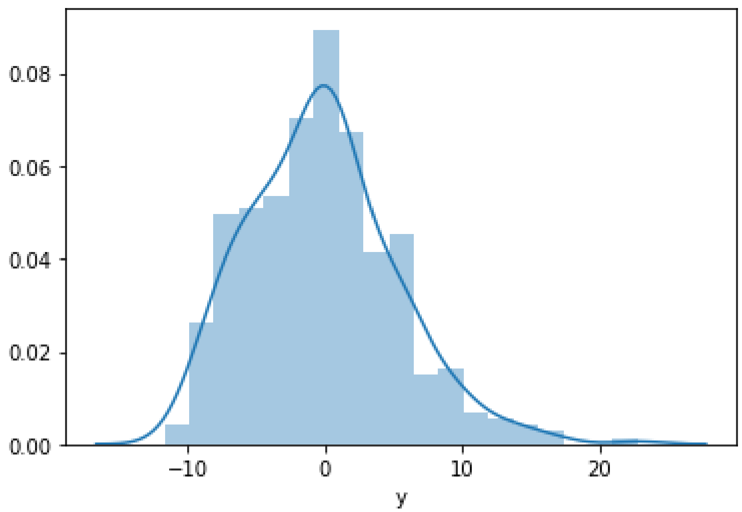
We can see we have 400 observations and 50 independent variables and 1 target variable y

Let's define x and y separately for modeling

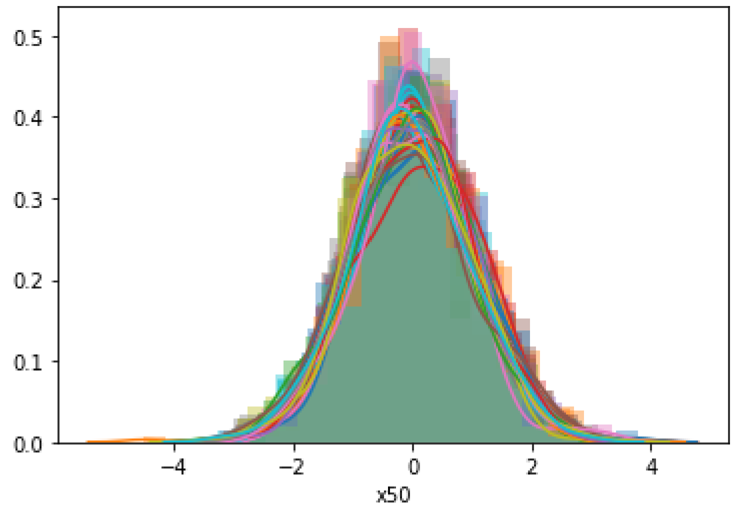
In [0]:
```
x=ExamData.iloc[:,1:]
y=ExamData.iloc[:,0]
```

In [14]:
```
sns.distplot(y)
```

Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff805af4898>



In [15]:
```
for i in range(50):
  sns.distplot(x.iloc[:,i])
```



So, our target variable is continuous with values ranging from -10 to 21 approximately. So, clearly this is a regression problem. Also, almost all predictors ranges from -4 to 4 approx so, we don't really need to scaling like standardization or normalization

We are using Train Test Split with ratio of 50% for one half to be trained and another half to be used to check the model accuracy

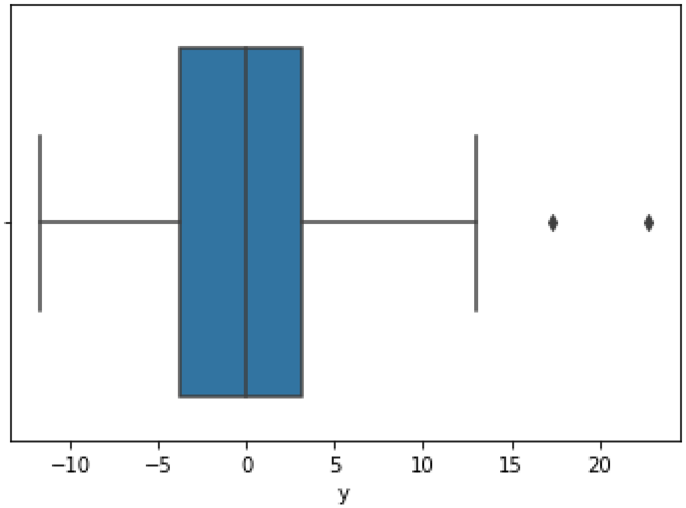
In [0]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.5, random_state=1)
```

Let's find the outliers by using boxplot method in training data and remove them so, that model doesn't learn from outlier values.

Let's check the outliers by having boxplot distribution of the target variable of trainig dataset only

In [17]:
```
sns.boxplot(y_train)
```

Out[17]: `<matplotlib.axes._subplots.AxesSubplot at 0x7ff8063aa9b0>`



We can see we have 2 outliers in the training y varaible which falls beyond the 1.5 times of interquartile range

Let's define the Interquartile range to detect the outliers

In [18]:
```
Q1 = y_train.quantile(0.25)
Q3 = y_train.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
6.881305133
```

In [19]:
```
 y_train[y_train > (Q3 + 1.5 * IQR)]
```

Out[19]:
```
243    17.289235
104    22.790085
Name: y, dtype: float64
```

We have 2 outliers which is beyond positive 1.5 times of IQR in y_train and thus we would want to remove these two observations from X and Y training so, that model doesn't learn from them

In [20]:
```
y_train[y_train < (Q1 - 1.5 * IQR)]
```

Out[20]:
```
Series([], Name: y, dtype: float64)
```

In [21]:
```
print(type(y_train),type(X_train))
```

```
<class 'pandas.core.series.Series'> <class 'pandas.core.frame.DataFrame'>
```

In [0]:
```
y_train.drop([104,243],inplace=True)
```

In [23]:
```
y_train.shape
```

Out[23]:
```
(198,)
```

In [24]:
```
X_train.drop([104,243],inplace=True)
```

```
/usr/local/lib/python3.6/dist-packages/pandas/core/frame.py:4117: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#re
turning-a-view-versus-a-copy
  errors=errors,
```

In [25]:
```
X_train.shape
```

Out[25]:
```
(198, 50)
```

In [0]:
```
from sklearn import preprocessing
X_train_scaled = preprocessing.StandardScaler().fit(X_train).transform(X_train)
```

Goal of this project is to reduce the Mean squared Error using different algorithms so, we would start with Null model and then proceed to simpler algorithms to complex ones

Null Model

In [27]:
```
#Null model, only a constant
Null_MSE=((y_test-y_train.mean())**2).mean()
print('Null model',Null_MSE)
```

```
Null model 28.907721505285107
```

We have Mean Squared Error for Null model that is when we have only constant terms and no variables at all is 28.90

Linear Model

In [28]:
```
#Linear model
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(X_train,y_train)
```

Out[28]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [0]:
```
y_pred= lm.predict(X_test)
```

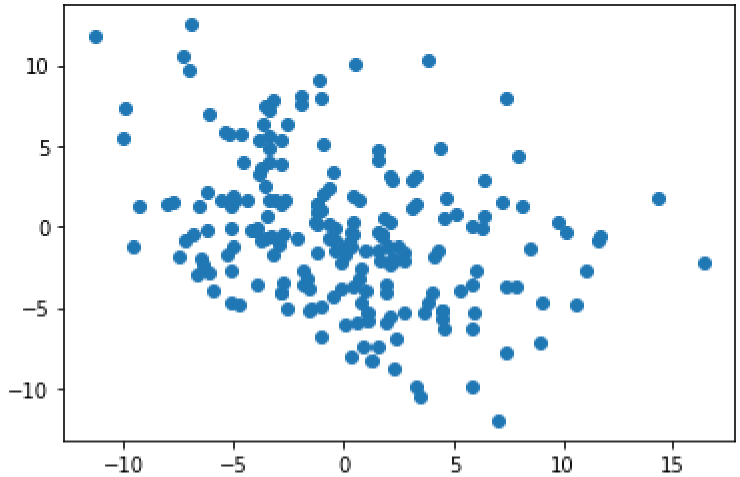
In [30]:
```
from sklearn import metrics
print('MSE:', metrics.mean_squared_error(y_test, y_pred))
```

```
MSE: 23.782570634177702
```

MSE decreased from 28.9 to 23.78 when we are using all the variables in Linear Regression

In [31]:
```
plt.scatter(y_test-y_pred,y_pred)
```

Out[31]: <matplotlib.collections.PathCollection at 0x7ff805cd26a0>

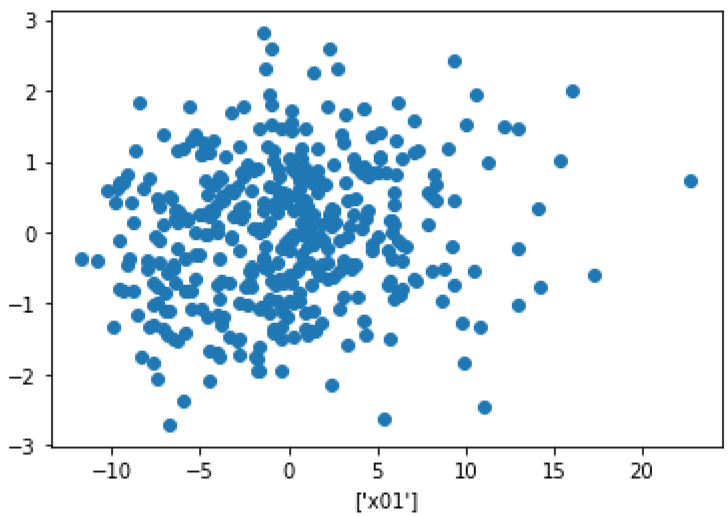


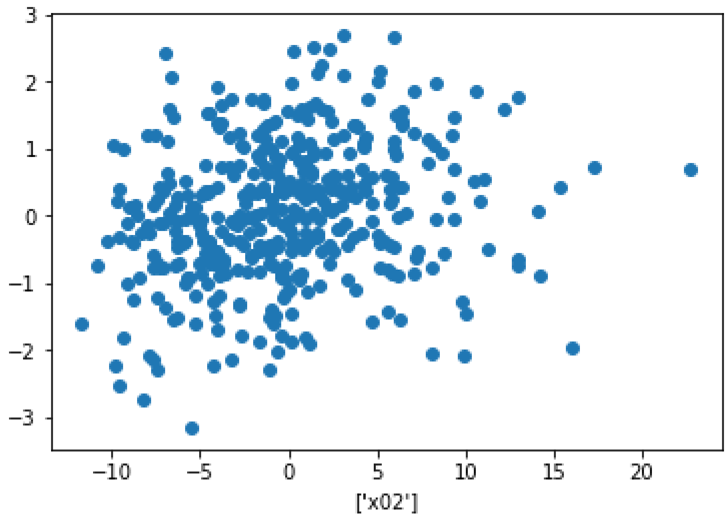
Since, there is no pattern, linear regression model is good fit for the data

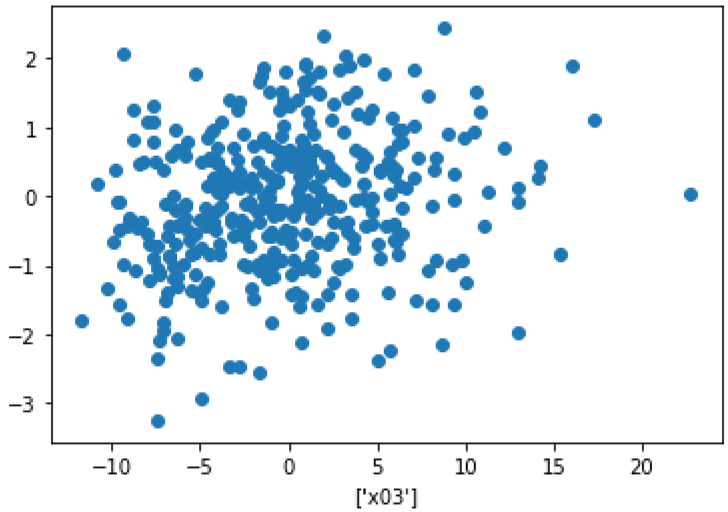
Let's see the plot of Y vs each of 50 predictors

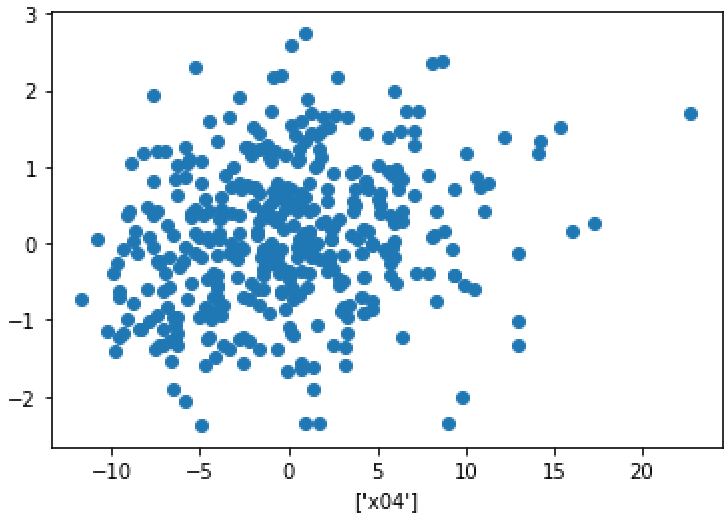
In [32]:
```
for i in x.columns:
    plt.figure()
    plt.scatter(ExamData['y'],ExamData[i])
    plt.xlabel([i])
```

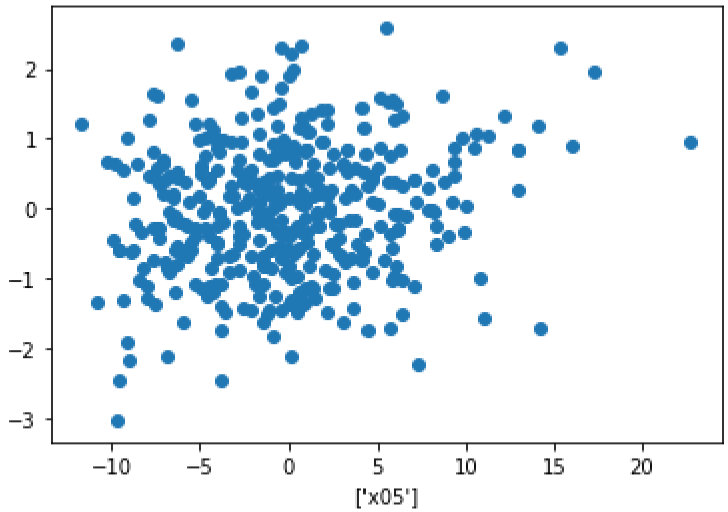
```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: RuntimeWarning: More than 20 figures have bee
n opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`) are retained until explic
itly closed and may consume too much memory. (To control this warning, see the rcParam `figure.max_open_warni
ng`).
```

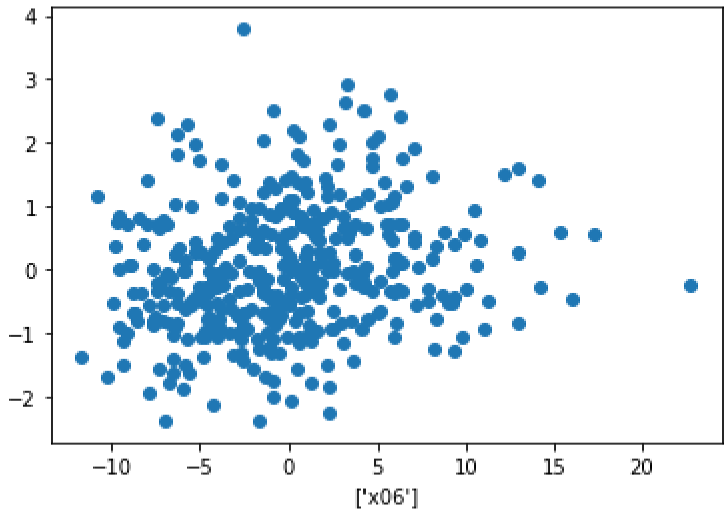


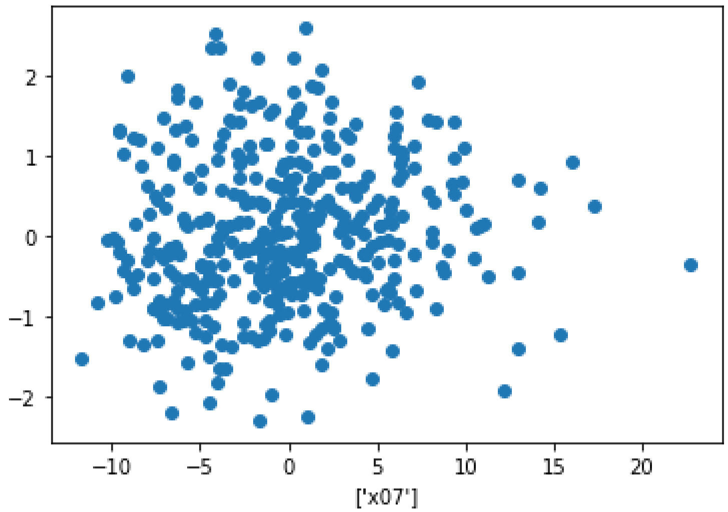


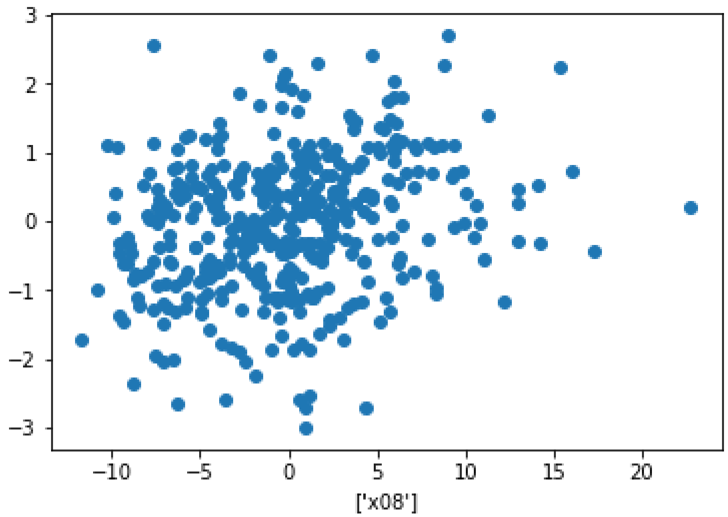

['x03']

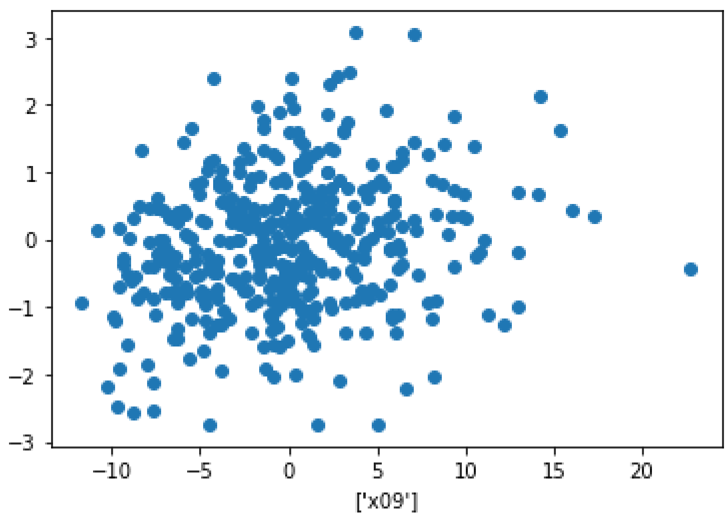

['x04']

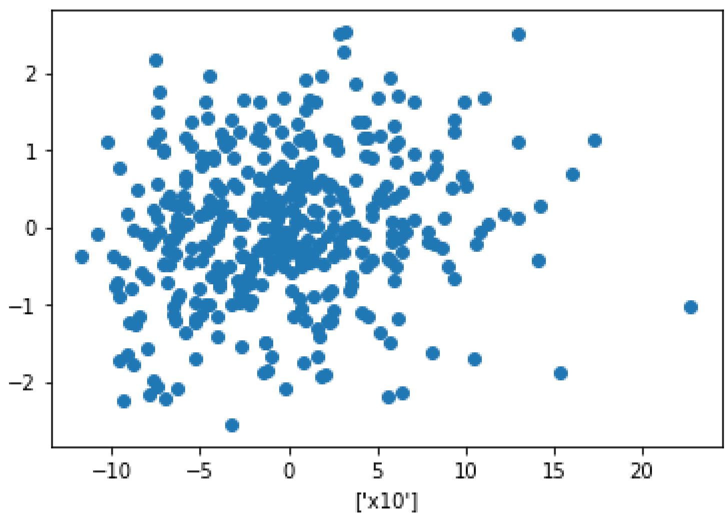

['x05']

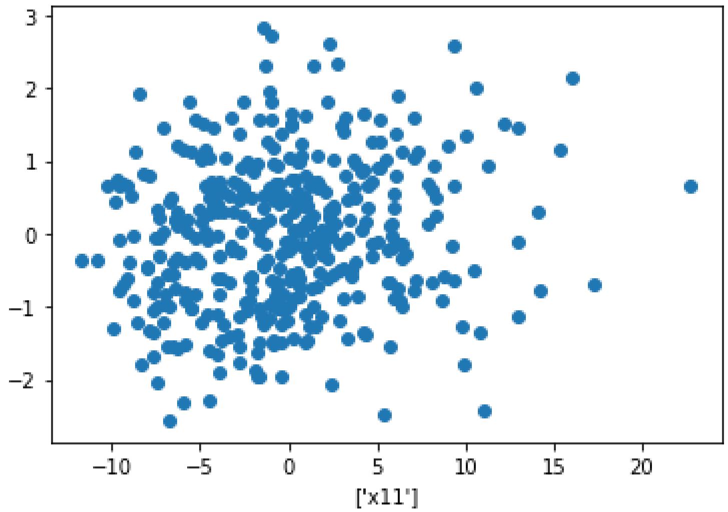

['x06']

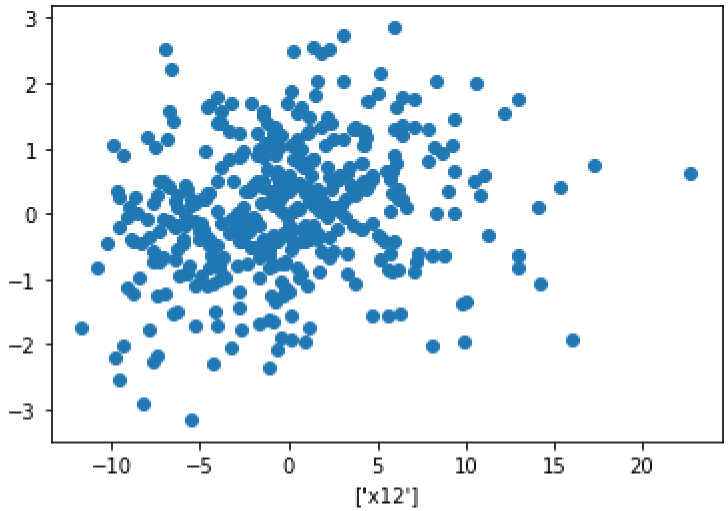

['x07']

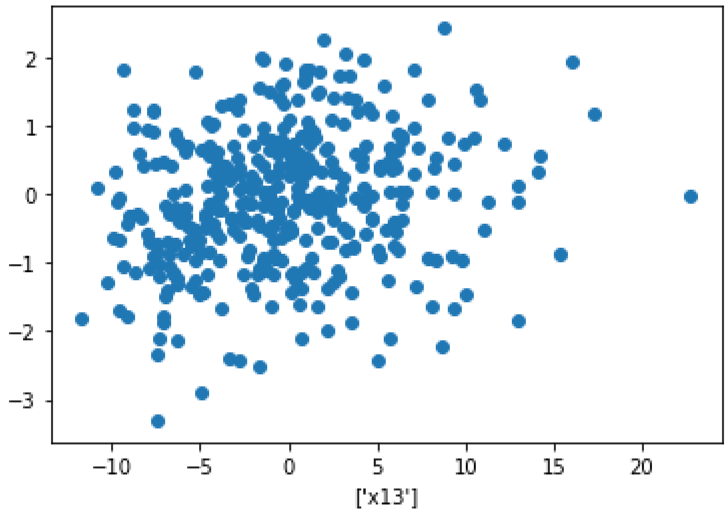

['x08']


3
2
1
0
−1
−2
−3
−10
−5
0
5
10
15
20
['x09']

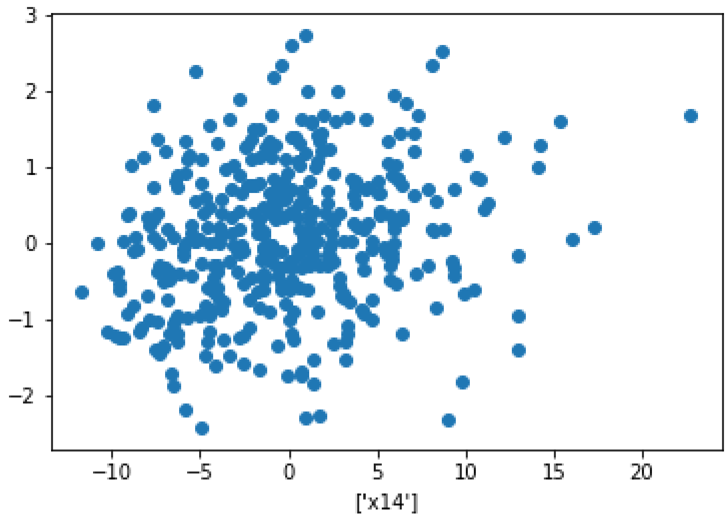

2
1
0
−1
−2
−10
−5
0
5
10
15
20
['x10']


3
2
1
0
−1
−2
−10
−5
0
5
10
15
20
['x11']

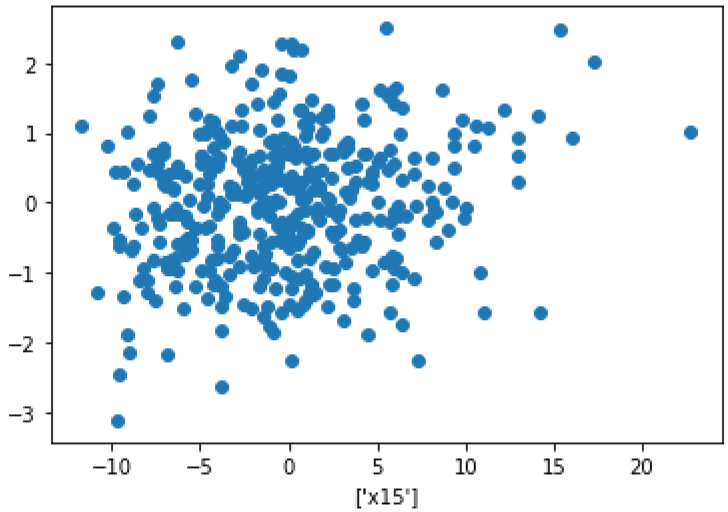

3
2
1
0
−1
−2
−3
−10
−5
0
5
10
15
20
['x12']

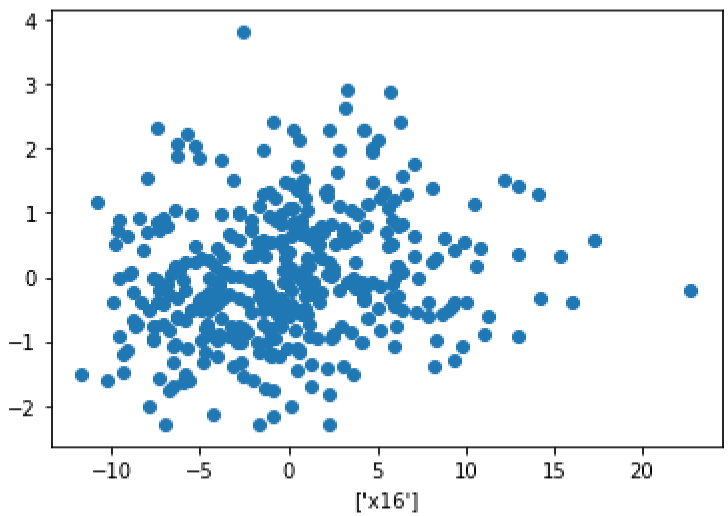


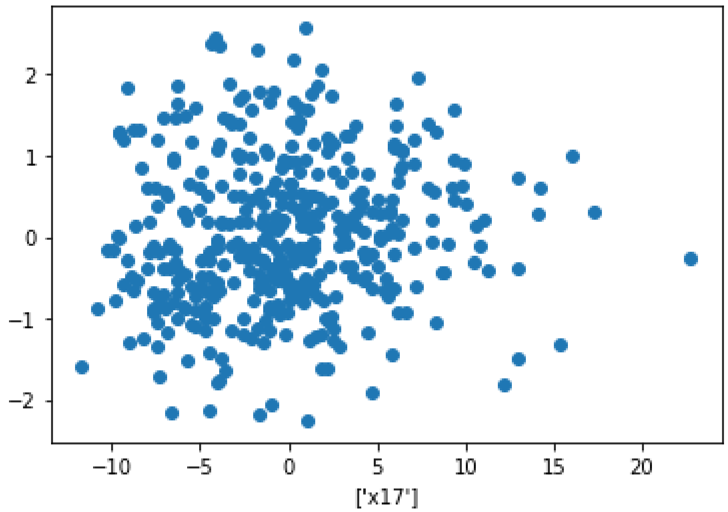
2
1
0
−1
−2
−3
−10
−5
0
5
10
15
20
['x13']



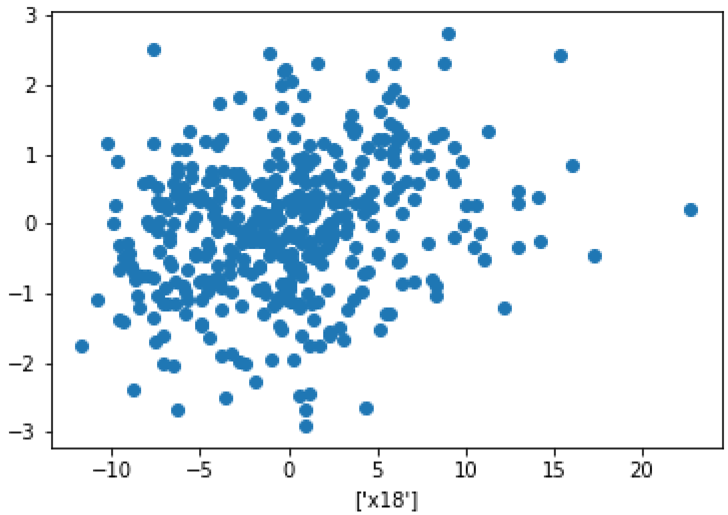
3
2
1
0
−1
−2
−10
−5
0
5
10
15
20
['x14']


['x15']

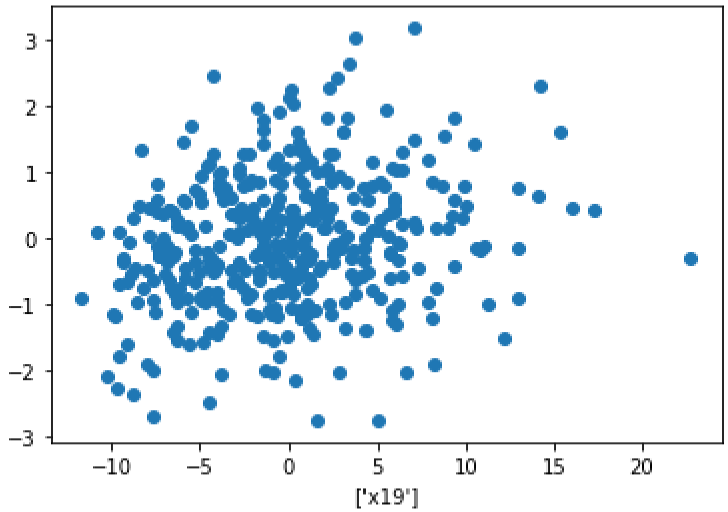

['x16']

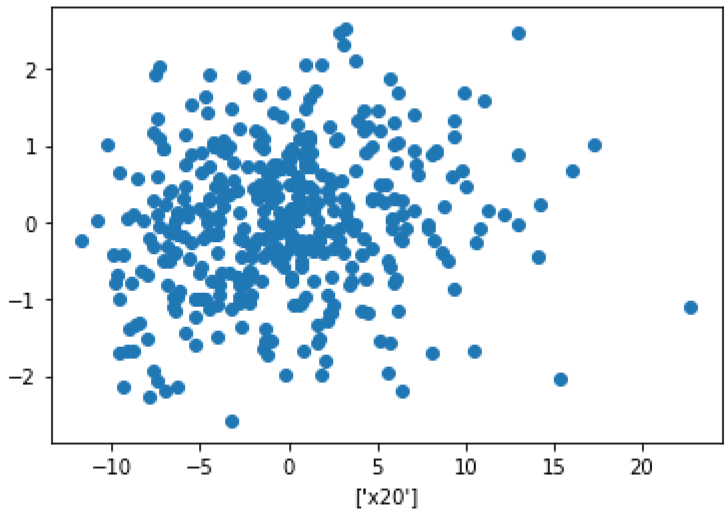


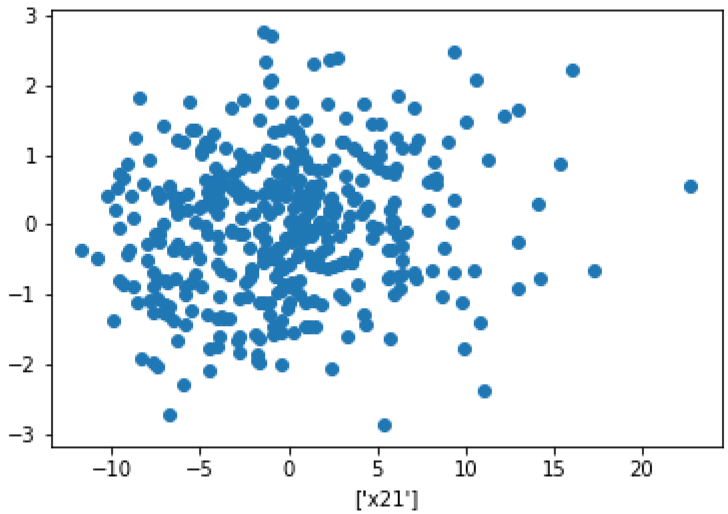
2
1
0
−1
−2
−10
−5
0
5
10
15
20
['x17']

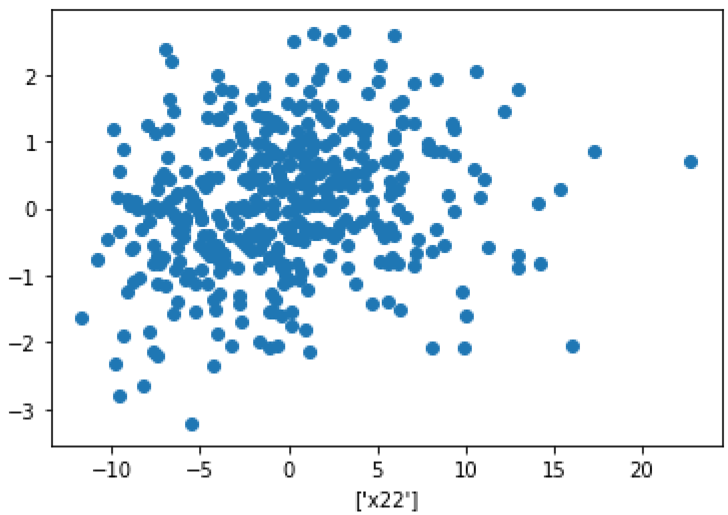


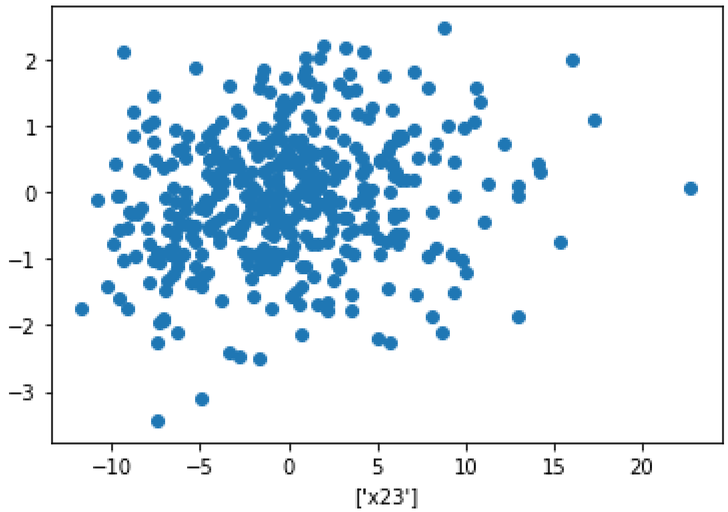
3
2
1
0
−1
−2
−3
−10
−5
0
5
10
15
20
['x18']

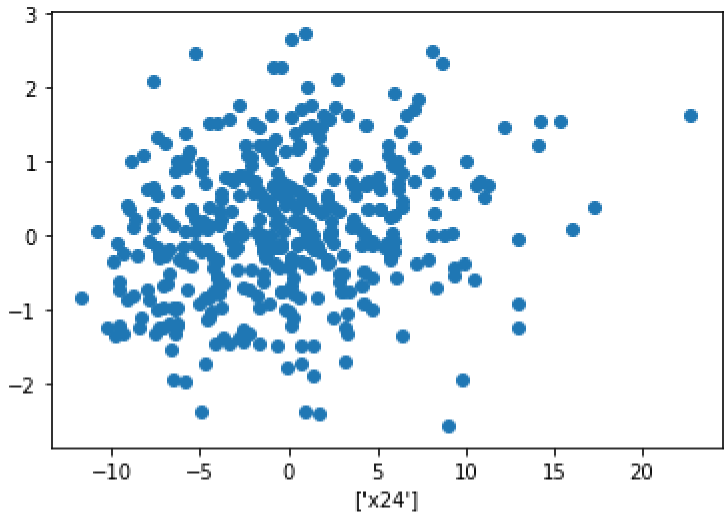

['x19']

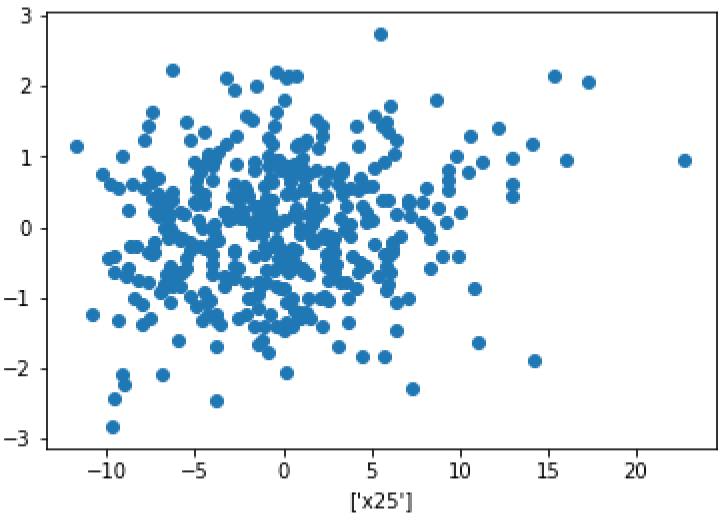

['x20']

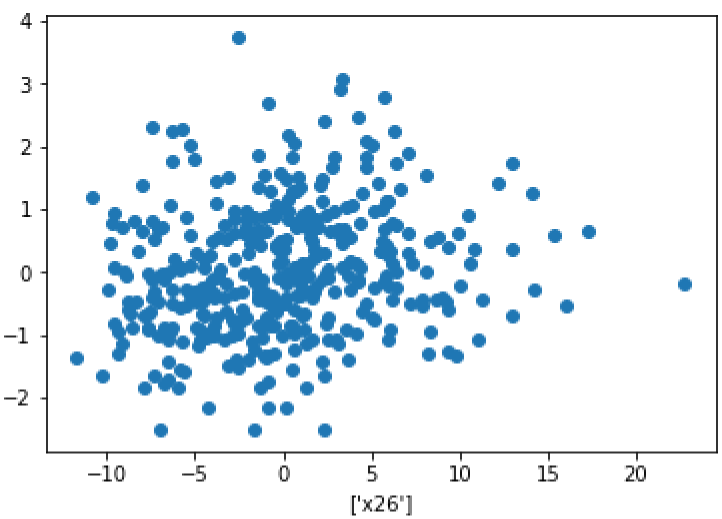

['x21']

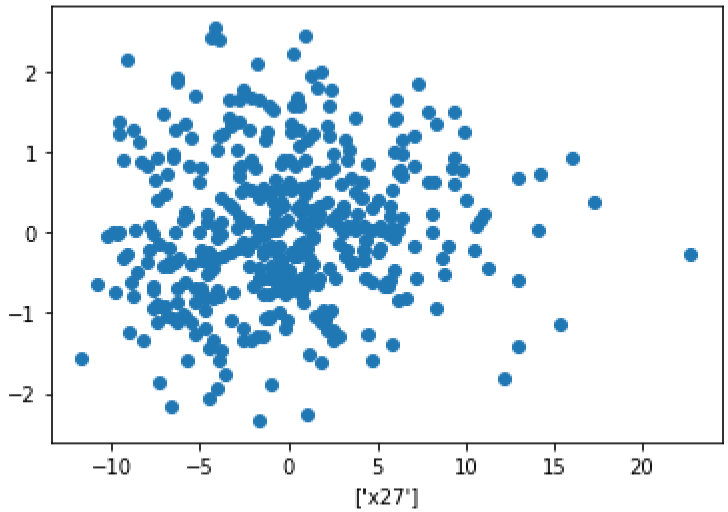

['x22']

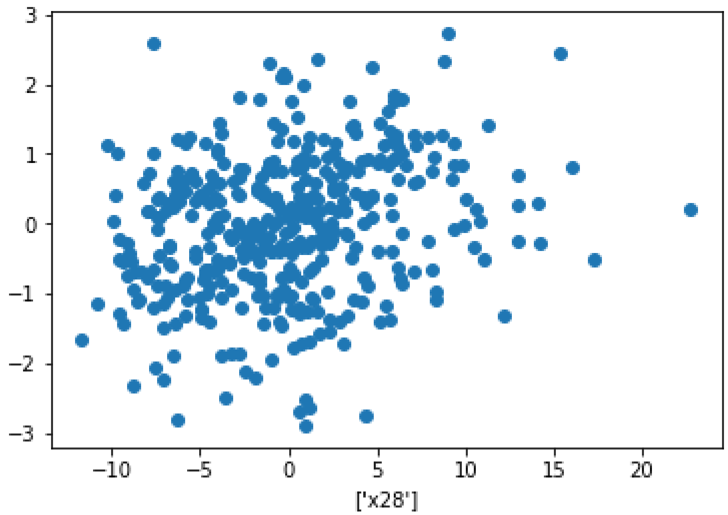

['x23']

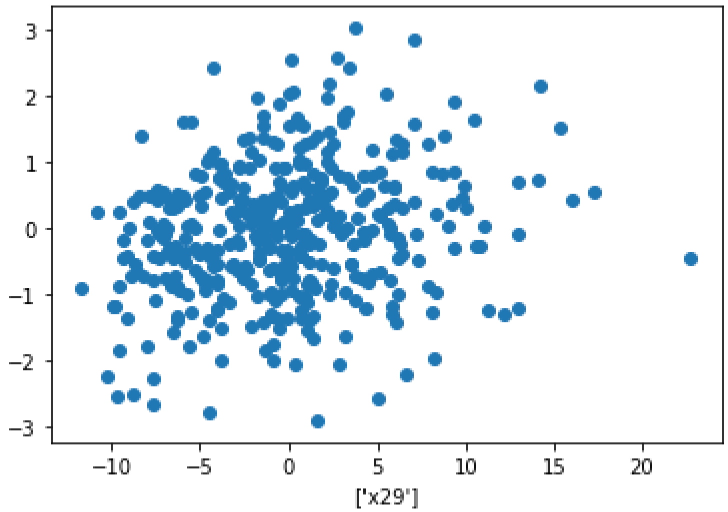

['x24']

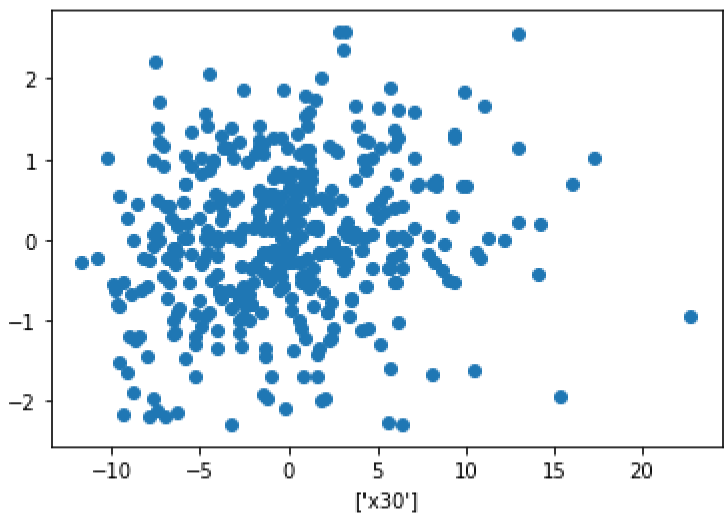

['x25']

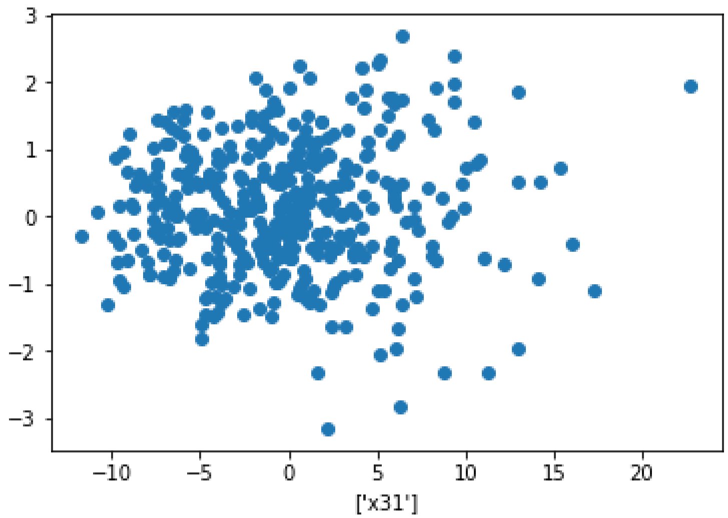

['x26']

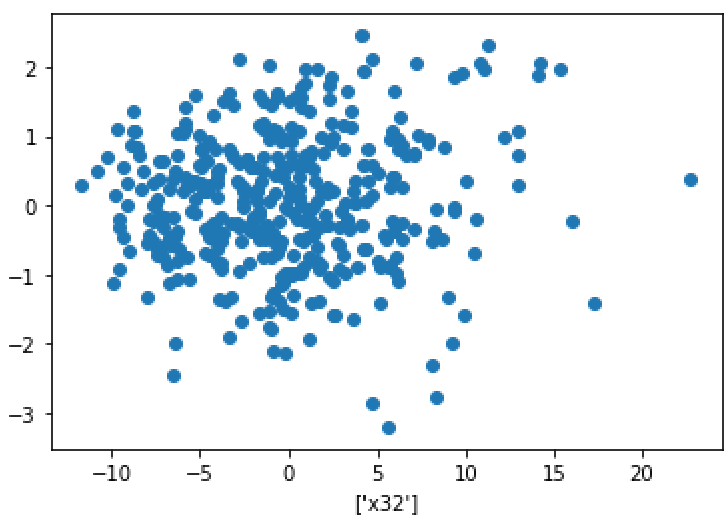

['x27']

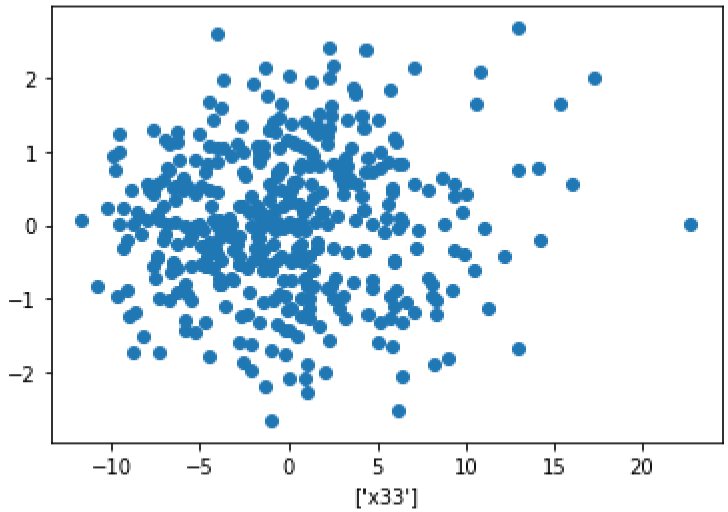

['x28']

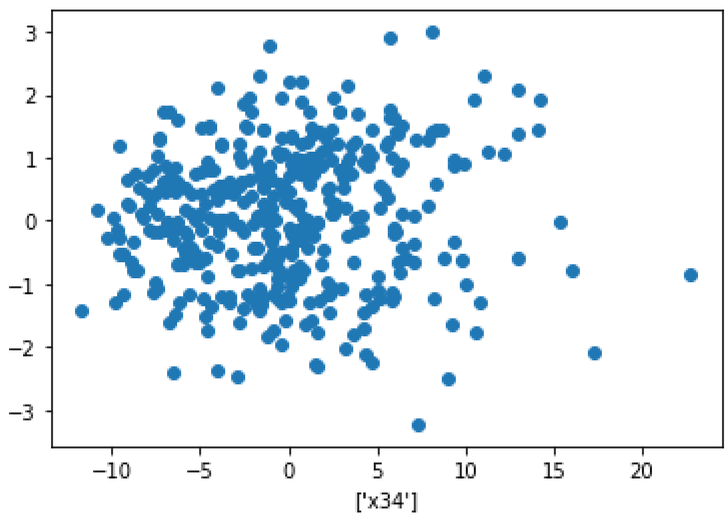

['x29']

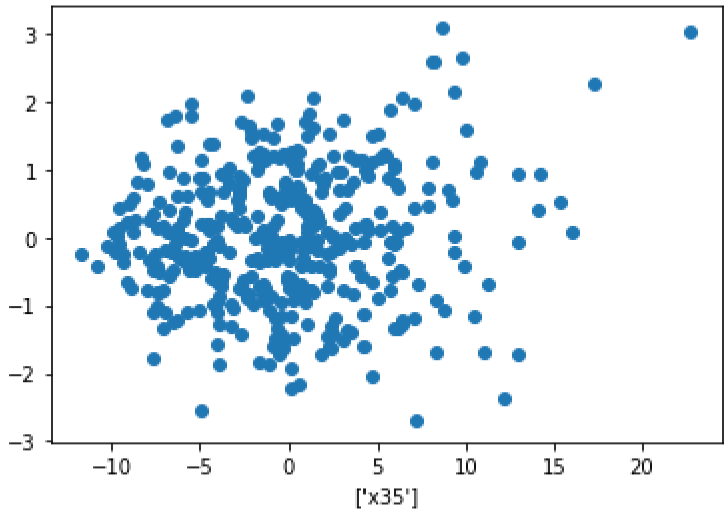

['x30']

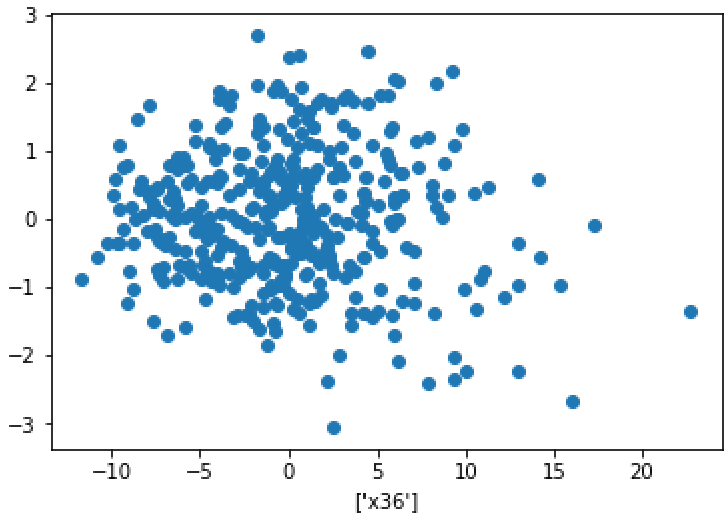

3
2
1
0
−1
−2
−3
−10 −5 0 5 10 15 20
['x31']

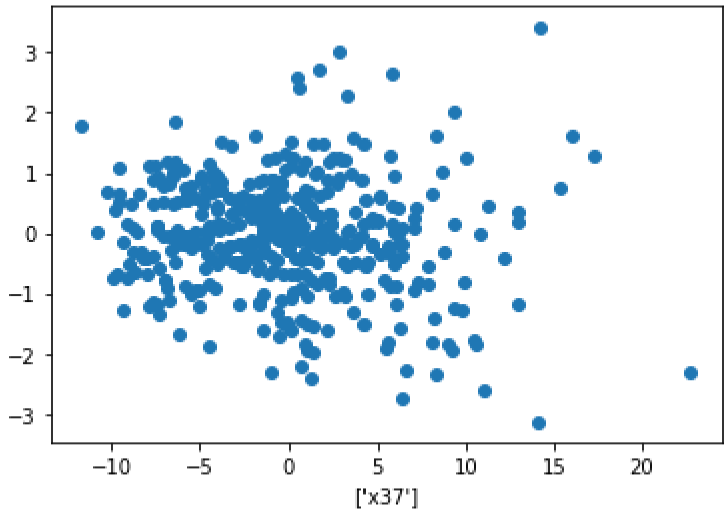

2
1
0
−1
−2
−3
−10 −5 0 5 10 15 20
['x32']


2
1
0
−1
−2
−10
−5
0
5
10
15
20
['x33']

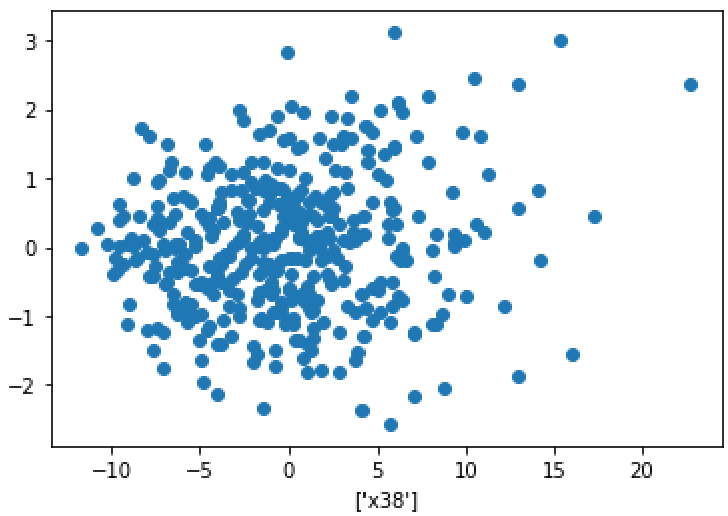

3
2
1
0
−1
−2
−3
−10
−5
0
5
10
15
20
['x34']


['x35']

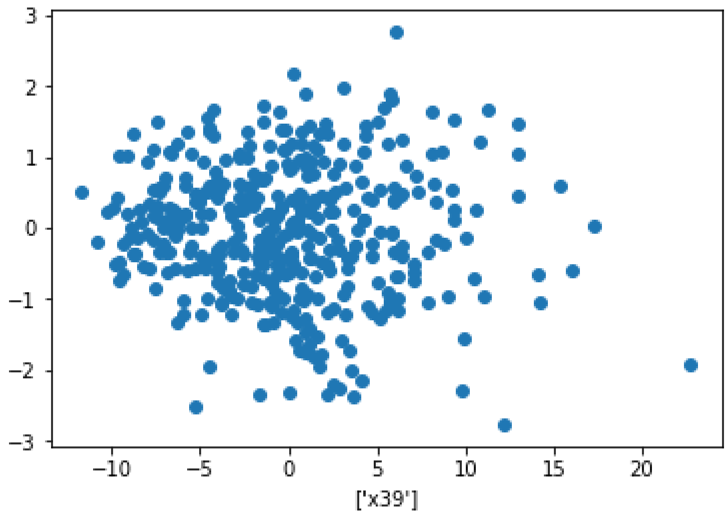

['x36']



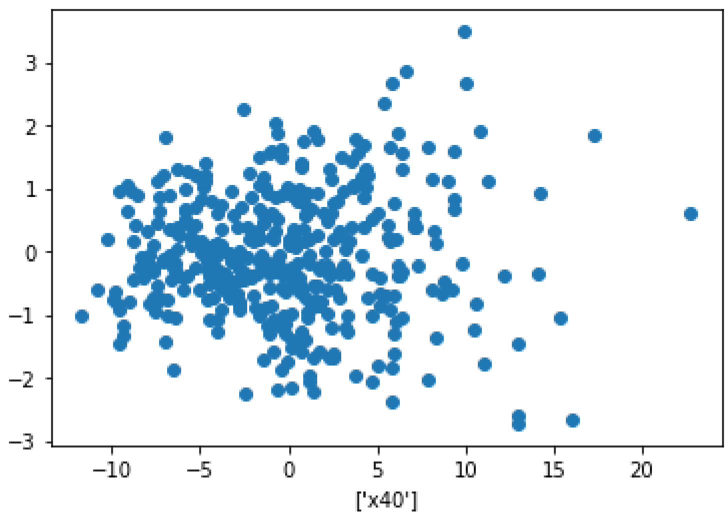
3
2
1
0
−1
−2
−3
−10
−5
0
5
10
15
20
['x37']



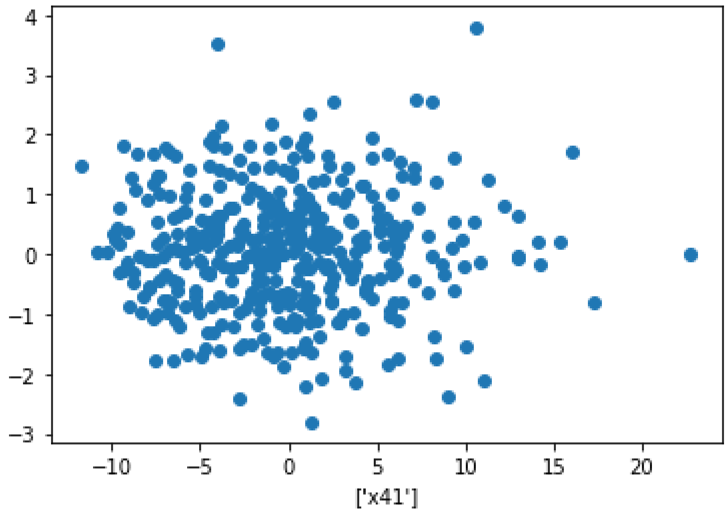
3
2
1
0
−1
−2
−10
−5
0
5
10
15
20
['x38']

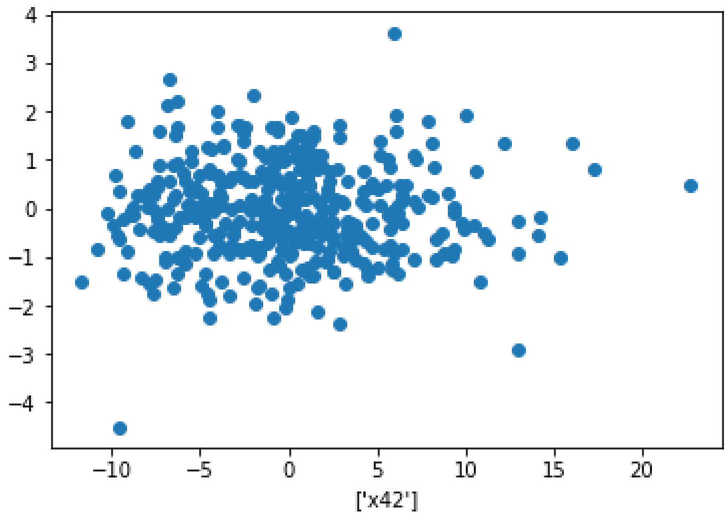


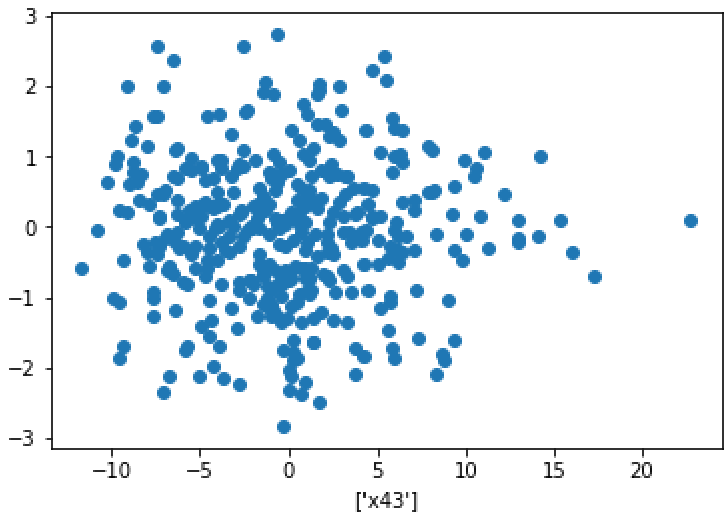
['x39']

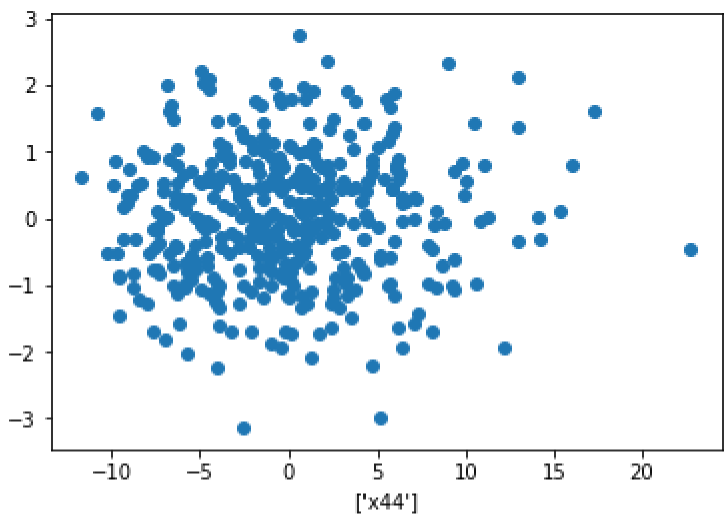


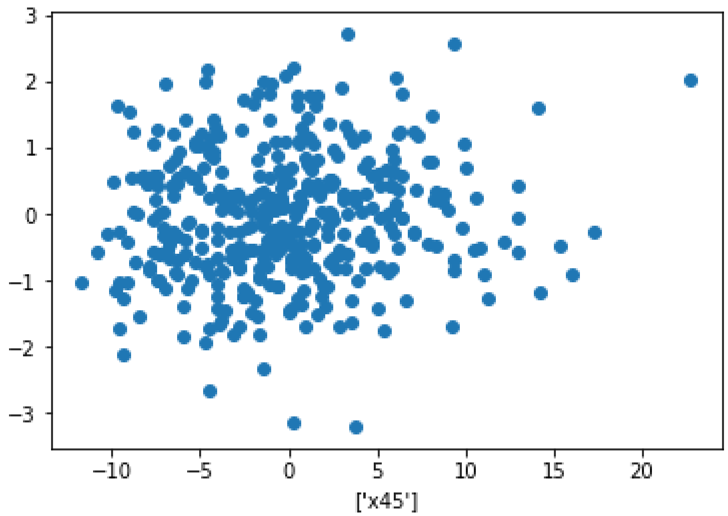
['x40']

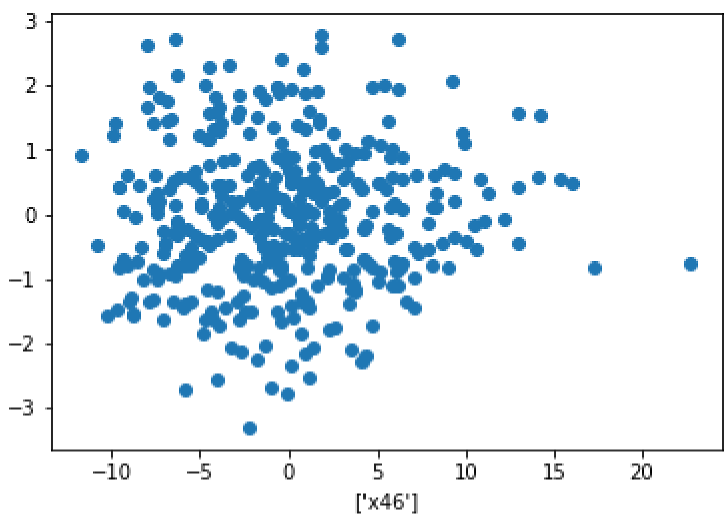

['x41']

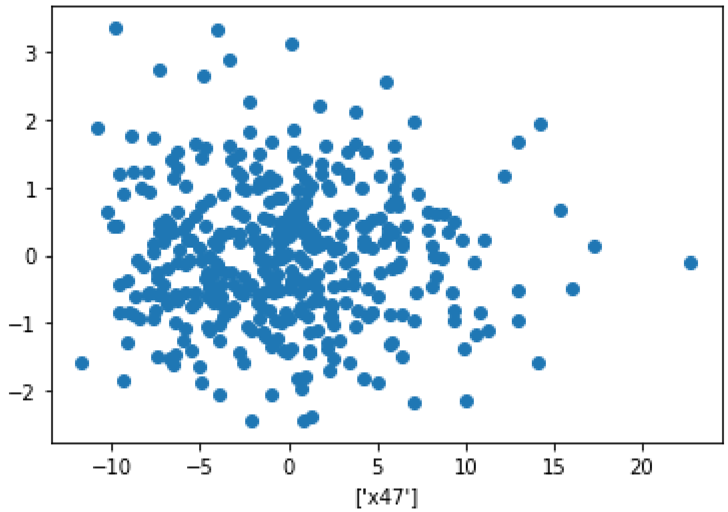

['x42']

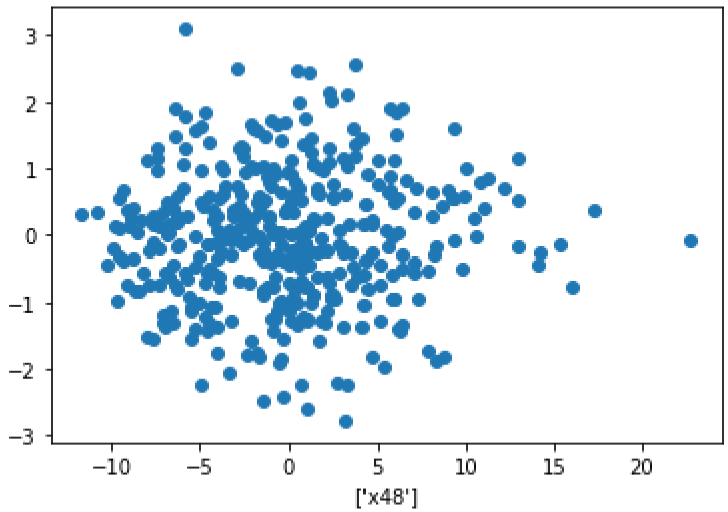

3
2
1
0
−1
−2
−3
−10
−5
0
5
10
15
20
['x43']

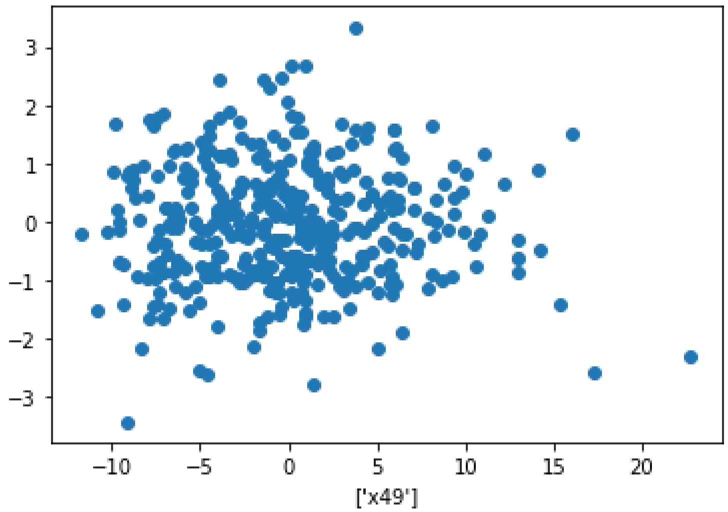

3
2
1
0
−1
−2
−3
−10
−5
0
5
10
15
20
['x44']

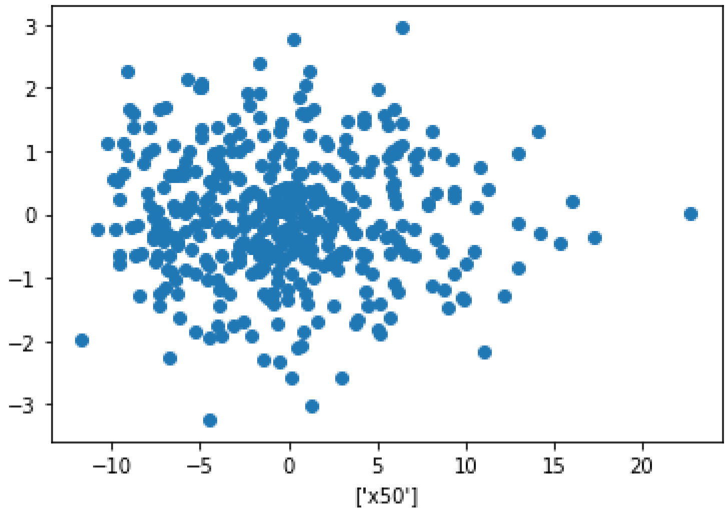

['x45']


['x46']


['x47']


['x48']





By seeing the plot of each x predictor and target variable y, it looks like each of the predictor contributes some information in the prediction of y

Let's see if there is multicollinearity among the predictors

In [0]:
```
#correlation_matrix=x.corr()
#correlation_matrix[correlation_matrix>0.7]
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [34]:
```
vf=[]
for i in range(x.shape[1]):
    vif=variance_inflation_factor(x.values,i)
    vf.append(vif)
print(vf)
```

```
[228.23739680781165, 208.6134404078362, 204.1758564663967, 216.8728092202751, 209.1648111033765, 218.93881421
883754, 203.213959240962, 205.11312020568704, 234.43372296688415, 190.52145081030676, 114.1277753920412, 107.
11963734135139, 123.7630227284136, 111.26405791459301, 111.50367521790403, 118.3957142378508, 108.74463396709
767, 111.66262742009825, 112.98989215900447, 95.12739600581183, 114.65341447558805, 119.16593260530172, 102.9
3091655994863, 106.82970713526485, 99.77598867444435, 121.73801668577076, 107.10447553095682, 99.219606517405
32, 111.30749806771529, 101.88780612337115, 1.1390460669950233, 1.1347702521924954, 1.2398367581072212, 1.122
9341263807588, 1.1707063537876892, 1.1077722466679816, 1.137746077467074, 1.095952137294243, 1.1065091588538
1, 1.1413268682287396, 1.129977400756082, 1.1407336726500947, 1.1502384524400628, 1.1870054829515753, 1.15554
27286054638, 1.1800239437090227, 1.1180221663554775, 1.1367699872530346, 1.1325418569843526, 1.18249156457189
23]
```

Since, if VIF is more than 10 then it represent multicollinearity. Let's see how many variables has VIF more than 10

In [35]:
```
vf=pd.DataFrame(vf)
vf[vf<10]
```

Out[35]:

	0
0	NaN
1	NaN
2	NaN
3	NaN
4	NaN
5	NaN
6	NaN
7	NaN
8	NaN
9	NaN
10	NaN
11	NaN
12	NaN
13	NaN
14	NaN
15	NaN
16	NaN
17	NaN
18	NaN
19	NaN
20	NaN
21	NaN
22	NaN

	0
23	NaN
24	NaN
25	NaN
26	NaN
27	NaN
28	NaN
29	NaN
30	1.139046
31	1.134770
32	1.239837
33	1.122934
34	1.170706
35	1.107772
36	1.137746
37	1.095952
38	1.106509
39	1.141327
40	1.129977
41	1.140734
42	1.150238
43	1.187005
44	1.155543
45	1.180024
46	1.118022

	0
47	1.136770
48	1.132542
49	1.182492

We can see in our variables, only x30 to x50 variables have low VIF around 1 thus no multicollinearity. We are going to use variables only from 30 to 50 in that case

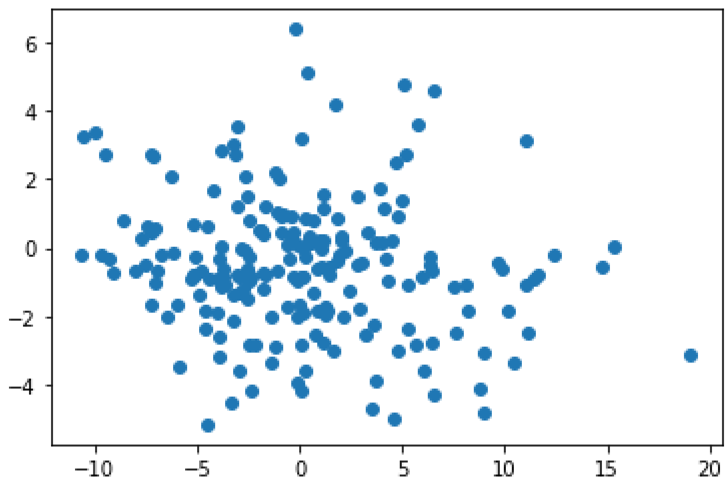
In [36]:
```
lm.fit(X_train.iloc[:,29:],y_train)
```
Out[36]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [37]:
```
y_pred_1= lm.predict(X_test.iloc[:,29:])
print('MSE:', metrics.mean_squared_error(y_test, y_pred_1))
```
```
MSE: 28.270774956447195
```

This MSE is worst than even the Linear Regression indicating that by losing the variables from x01 to x29, we are losing much information of y

In [38]:
```
plt.scatter(y_test-y_pred_1,y_pred_1)
```

Out[38]: <matplotlib.collections.PathCollection at 0x7ff805ed84e0>



Let's fit the same linear model but with first 29 variables instead of last 31 variables and see how model performs

In [39]:
```
x_2=X_train.iloc[:,:29]
lm.fit(x_2,y_train)
```

Out[39]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

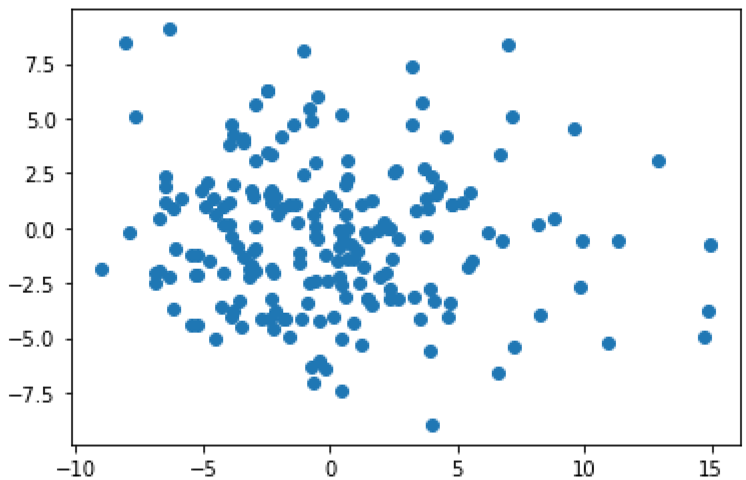
In [40]:
```
y_pred_2= lm.predict(X_test.iloc[:,:29])
print('MSE:', metrics.mean_squared_error(y_test, y_pred_2))
```

```
MSE: 20.50058173101595
```

We managed to decrease the MSE to 20.5 incidating that first 29 variables has more information of y than 2nd set of 21 variables

In [41]:
```
plt.scatter(y_test-y_pred_2,y_pred_2)
```

Out[41]: <matplotlib.collections.PathCollection at 0x7ff8059f4be0>



Again since there is no clear pattern, linear regression seems to be a good fit for the data

Quadratic Model

In [42]:
```
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree = 2)
X_poly = poly.fit_transform(X_train)

poly.fit(X_poly, y_train)
lm.fit(X_poly, y_train)
```

Out[42]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

In [43]:
```
y_pred_3= lm.predict(poly.fit_transform(X_test))
print('MSE:', metrics.mean_squared_error(y_test, y_pred_3))
```

```
MSE: 18.733680182554075
```

Let's try to fit the quadratic model on selected 1st 29 features and see if perfoance improves or not.

In [44]:
```
X_poly = poly.fit_transform(x_2)

poly.fit(X_poly, y_train)
lm.fit(X_poly, y_train)
```

Out[44]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [46]:
```
y_pred_quad_set= lm.predict(poly.fit_transform(X_test.iloc[:,:29]))
print('MSE:', metrics.mean_squared_error(y_test, y_pred_quad_set))
```

```
MSE: 95.93603207246629
```

MSE decreases to even lower number of 95.93 indicating we are losing information

We can see MSE decreased to 18.733% indicating quadratic model is better fit

Decision Tree Regression

In [0]:
```
#Let's do Decision Tree regression
from sklearn.tree import DecisionTreeRegressor
```

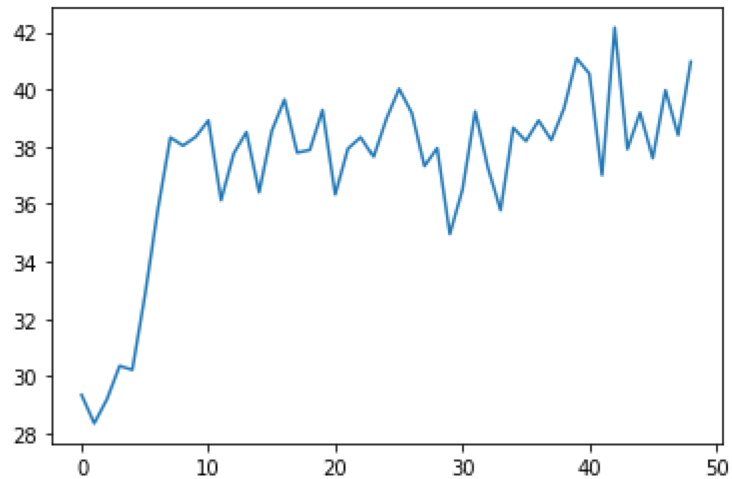
In [0]:
```
Acc_Decision=[]
for i in range(1,50):
  regr_1 = DecisionTreeRegressor(max_depth=i)
  regr_1.fit(X_train, y_train)
  y_pred_4 = regr_1.predict(X_test)
  MSE=metrics.mean_squared_error(y_test, y_pred_4)
  Acc_Decision.append(MSE)
```

In [49]:
```
min(Acc_Decision)
```

Out[49]: 28.36175947896122

In [50]:
```
plt.plot(Acc_Decision)
```

Out[50]: [<matplotlib.lines.Line2D at 0x7ff805f1c7f0>]



We tried to use the Decision Tree regression perormance with depth ranging from 1 to 50, is not that great and is infact worst than the Linear regression also. Our aim is to reduce MSE by applying different modeling methods

Random Forest Regression

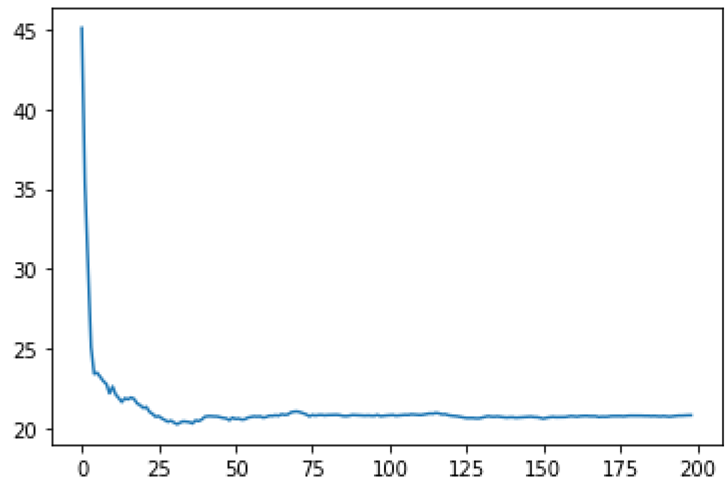
In [0]:
```
from sklearn.ensemble import RandomForestRegressor
Acc_Random=[]
for i in range(1,200):
  regressor = RandomForestRegressor(n_estimators = i, random_state = 0)
  regressor.fit(X_train, y_train)
  y_pred_5=regressor.predict(X_test)
  MSE=metrics.mean_squared_error(y_test, y_pred_5)
  Acc_Random.append(MSE)
```

In [52]:
```
min(Acc_Random)
```

Out[52]:
```
20.279687867097145
```

In [53]:
```
plt.plot(Acc_Random)
```

Out[53]:
```
[<matplotlib.lines.Line2D at 0x7ff7fe157c50>]
```



We can see the MSE has reached the minimum MSE of 20.279 and remains constant even if number of decreased

Ridge Regression

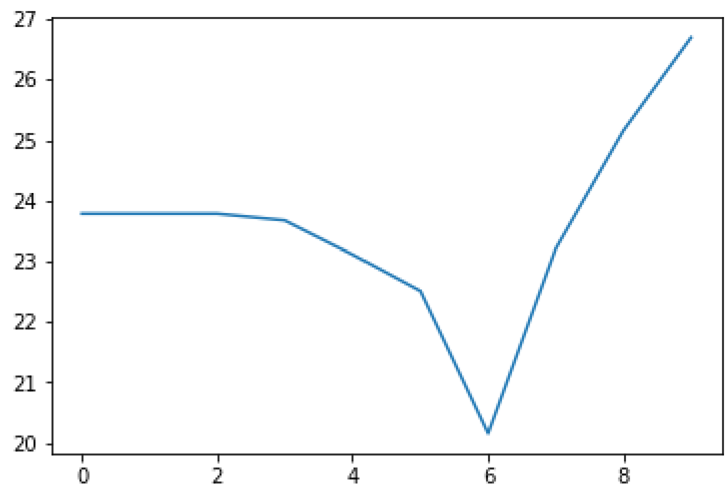
In [0]:
```
from sklearn.linear_model import Ridge
Acc_Ridge=[]
for i in [1e-15, 1e-10, 1e-8, 1e-4, 1e-3,1e-2, 1, 5, 10, 20]:
  ridge=Ridge(alpha=i,normalize=True)
  ridge.fit(X_train,y_train)
  y_pred_6=ridge.predict(X_test)
  MSE=metrics.mean_squared_error(y_test, y_pred_6)
  Acc_Ridge.append(MSE)
```

In [55]:
```
min(Acc_Ridge)
```

Out[55]: 20.158611518654055

In [56]:
```
plt.plot(Acc_Ridge)
```

Out[56]: [<matplotlib.lines.Line2D at 0x7ff7fe7c67b8>]



Thus Ridge regression has minimum MSE of 20.158 when penalty tuning parameter is 1. Please note that Ridge Regression uses L2 norm to panalize the predictor's cofficient thus can't be used to do variable selection

In [57]:
```
from sklearn.model_selection import GridSearchCV
parameters={'alpha':[1e-15, 1e-10, 1e-8, 1e-4, 1e-3,1e-2, 1, 5, 10, 20]}
ridge=Ridge()
ridge_regressor=GridSearchCV(ridge,parameters,scoring='neg_mean_squared_error',cv=2)
ridge_regressor.fit(x,y)
```

Out[57]:
```
GridSearchCV(cv=2, error_score='raise-deprecating',
             estimator=Ridge(alpha=1.0, copy_X=True, fit_intercept=True,
                             max_iter=None, normalize=False, random_state=None,
                             solver='auto', tol=0.001),
             iid='warn', n_jobs=None,
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.0001, 0.001, 0.01, 1,
                                   5, 10, 20]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring='neg_mean_squared_error', verbose=0)
```

In [58]:
```
print(ridge_regressor.best_params_, ridge_regressor.best_score_)
```

```
{'alpha': 20} -20.571701795778154
```

This Ridge Regression using Grid Search using 2-Fold cross validation is same as Ridge regression on 50% train test split and thus has almost same MSE

Lasso Regression

In [0]:
```
from sklearn.linear_model import Lasso
from sklearn.feature_selection import SelectFromModel
from sklearn.preprocessing import StandardScaler
```

In [60]:
```
scaler = StandardScaler()
scaler.fit(X_train)
```

Out[60]:
```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

Let's do the variable selection using LASSO Regression. Lasso uses L1 norm to penalize predictors and thus it can be used to select variables.

In [61]:
```
sel_ = SelectFromModel(LinearRegression())
sel_.fit(scaler.transform(X_train),y_train)
```

Out[61]:
```
SelectFromModel(estimator=LinearRegression(copy_X=True, fit_intercept=True,
                                           n_jobs=None, normalize=False),
                max_features=None, norm_order=1, prefit=False, threshold=None)
```

In [62]:
```
sel_.get_support()
```

Out[62]:
```
array([ True, False,  True,  True,  True,  True,  True,  True, False,
        True,  True,  True,  True, False, False,  True, False,  True,
        True, False,  True,  True,  True, False,  True, False,  True,
       False, False,  True, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False])
```

Those with True values are selected variables by Lasso and one with False has been reduced to 0

Let's try to build the model using selected Features set

In [0]:
```
feauture_selected=pd.DataFrame(sel_.get_support(),columns=['feature'])
[feauture_selected['feature']==True]
feature_set=['x01','x02','x03','x04','x05','x06','x07','x09','x10','x11','x12','x15','x17','x18','x20','x21',
'x22','x24','x26','x29']
x_selected=x[feature_set]
```

In [65]:
```
X_train_sel, X_test_sel, y_train_sel, y_test_sel = train_test_split(x_selected, y, test_size=0.5, random_stat
e=1)

lm.fit(X_train_sel, y_train_sel)
```

Out[65]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [66]:
```
y_pred_sel= lm.predict(X_test_sel)
print('MSE:', metrics.mean_squared_error(y_test_sel, y_pred_sel))
```

```
MSE: 20.76247077216838
```

Thus MSE has decreased to 20.76 which is quite improvement but not much. which indicates we have almost same information as Ridge or Linear Regression on first 29 variables. Compared to Linear Regression on all the 50 variables, we had MSE of 23.77 and we are minimizing MSE by decreasing the number of variables from 50 to 20 which is great improvement

In [0]:
```
#Random Forest on selected features

Acc_Random_sel=[]
for i in range(1,200):
  regressor = RandomForestRegressor(n_estimators = i, random_state = 0)
  regressor.fit(X_train_sel, y_train_sel)
  y_pred_sel_random=regressor.predict(X_test_sel)
  MSE=metrics.mean_squared_error(y_test_sel, y_pred_sel_random)
  Acc_Random_sel.append(MSE)
```

In [68]:
```
min(Acc_Random_sel)
```

Out[68]:
```
24.443442944963035
```

Random Forest on selected Features of 20 predictors is even worst than the random Forest on all 50 features indicatig we are losing information

Let's use the Lasso Regression to predict Y and see it's MSE instead of variable selection. Here number of iterations to converge the parameters is 100000

In [69]:
```
for i in [1e-15, 1e-10, 1e-8, 1e-4, 1e-3,1e-2, 1, 5, 10, 20]:
  model_lasso = Lasso(alpha=i,max_iter = 100000)
  model_lasso.fit(X_train, y_train)
  pred_test_lasso= model_lasso.predict(X_test)
  print(metrics.mean_squared_error(y_test,pred_test_lasso))
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/coordinate_descent.py:475: ConvergenceWarning: Ob
jective did not converge. You might want to increase the number of iterations. Duality gap: 482.462601424491
5, tolerance: 0.5062479822606639
  positive)
```

```
23.782570634178445
23.782570598487528
23.782567066252813
23.747118911414262
23.454660531394467
22.381107314800232
27.480870793917646
28.907721505285117
28.907721505285117
28.907721505285117
```

MSE isn't decreaing much on even 100000 iterations. This performance is worst than even Ridge Regression

Let's build a Neural Network model using Keras. This would be Sequantial Feed Forward Neural Network model with activation fucntion as Linear since, this is Regression problem. Loss function used is Mean Squared Error and optmization method used is Stochastic Gradient Descent.

Neural Network Model

Let's import all the libraries to build the model

In [70]:
```
#Neural Network model
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.layers import LSTM
from keras.callbacks import Callback
from keras.models import Sequential
from keras.layers import LSTM, Dense, Activation
import pickle
from mpl_toolkits.mplot3d import Axes3D
import sys
from queue import Queue
import json
%matplotlib inline
```

```
Using TensorFlow backend.
```

The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade (https://www.tensorflow.org/guide/migrate) now or ensure your notebook will continue to use TensorFlow 1.x via the `%tensorflow_version 1.x` magic: more info (https://colab.research.google.com/notebooks/tensorflow_version.ipynb).

Let's use a class to see how Loss is being reduced with each learning from each batch of 72

In [0]:
```
class LossHistory(Callback):
    def on_train_begin(self, logs={}):
        self.losses = []

    def on_batch_end(self, batch, logs={}):
        sys.stdout.write(str(logs.get('loss'))+str(', '))
        sys.stdout.flush()
        self.losses.append(logs.get('loss'))

lr = LossHistory()
```

Let's define the structure of Neural Network model. Since, we have 50 predictors so, outside layer would have 50 neurons. We would define one hidden layer with 20 neurons and since, there is one y target variable and is a regression problem, output layer containing one 1 neurons. Since, our data is small and has only 200 observations (Note that we are removing outliers in case of Neural Network model from training datset since, it is capable of learning from outliers by backpropagation), we have used epoch size of 1000

In [0]:
```
number_of_neurons_layer1 = 50
number_of_neurons_layer2 = 20
number_of_neurons_layer3 = 1
number_of_epochs = 1000
```

In [0]:
```
dim =  50
samples = 200
```

In [0]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.5, random_state=1)
```

In [0]:
```
# design network
from keras import optimizers
sgd = optimizers.SGD(lr=0.01, clipnorm=1.)

model = Sequential()
model.add(Dense(number_of_neurons_layer1,input_shape=(dim, ), activation='linear'))
model.add(Dense(number_of_neurons_layer2, activation='linear'))
model.add(Dense(number_of_neurons_layer3, activation='linear'))
model.compile(loss='mean_squared_error', optimizer=sgd)

def train(data,label):
    model.fit(data, label, epochs=number_of_epochs, batch_size=72, validation_data=(data, label), verbose=0,
shuffle=True,callbacks=[lr])

def score(data):
    return model.predict(data)
```

In [78]:
```
train(X_train,y_train)
```

25.28822, 32.1681, 37.339314, 24.857685, 30.013388, 38.246815, 29.575705, 32.01464, 27.468153, 25.548523, 40.
93689, 18.960724, 35.34854, 21.070946, 29.872519, 23.434183, 27.07907, 35.426914, 27.727484, 27.457832, 27.55
5311, 21.745632, 24.24398, 37.393536, 28.835226, 24.141514, 26.603512, 31.350393, 17.59992, 30.031834, 35.537
86, 19.260353, 20.761034, 31.00826, 23.51952, 19.458273, 20.9521, 32.92854, 18.75256, 25.415955, 26.538483, 1
9.706142, 24.99003, 17.135939, 30.784842, 31.833366, 19.868925, 17.057491, 21.992432, 18.70217, 29.90968, 17.
260132, 27.653086, 23.200886, 28.22744, 18.20826, 19.861288, 20.303488, 22.958263, 22.573927, 19.684753, 17.9
52837, 28.663464, 22.853241, 25.360779, 13.974365, 25.718283, 21.264387, 14.489476, 29.059095, 11.932207, 21.
189932, 18.39994, 16.201057, 28.429544, 18.492886, 22.680122, 19.020939, 19.134045, 19.273468, 21.577885, 19.
891323, 18.80097, 20.24629, 22.39971, 16.02381, 19.663235, 21.13102, 18.410482, 17.371801, 19.691862, 18.9743
75, 17.650358, 20.407982, 15.874342, 19.948532, 22.39183, 17.297995, 14.784863, 14.791033, 17.68857, 23.35785
3, 16.15442, 19.482445, 18.576204, 14.210124, 24.66663, 13.824416, 16.446344, 13.409368, 24.850973, 24.51871
9, 13.535472, 13.6883, 17.562191, 18.162384, 16.15308, 22.638712, 13.148387, 15.495734, 15.44133, 14.94167, 2
1.975584, 19.77463, 16.817987, 13.531019, 22.158634, 13.102633, 14.835672, 16.638792, 15.138546, 18.922169, 1
8.634054, 16.580116, 14.097758, 19.320665, 14.380768, 15.68226, 20.284458, 14.704885, 13.708962, 14.241448, 1
7.880638, 17.158838, 13.634084, 19.70151, 15.236677, 14.808332, 18.029213, 15.644937, 15.960829, 13.173888, 2
0.127008, 12.959908, 18.734016, 16.601772, 15.745268, 16.502607, 15.60832, 14.26689, 19.359253, 13.624877, 1
5.284496, 18.94095, 12.715173, 12.697978, 18.190912, 16.796577, 17.814562, 14.648424, 14.602107, 15.935057, 1
1.929655, 20.383932, 15.494361, 14.658617, 17.264967, 12.374532, 21.47721, 12.369688, 13.695973, 16.10254, 1
7.461006, 18.938826, 15.006707, 11.984602, 17.456757, 14.959032, 13.877481, 14.413622, 19.098854, 12.417058,
18.78262, 11.88676, 15.828282, 14.009479, 17.073212, 15.233498, 13.335844, 15.097946, 18.519785, 13.1562195,
14.459797, 19.461882, 16.052177, 19.000751, 9.825661, 16.29626, 17.205925, 11.736868, 13.395454, 13.841223, 1
9.752995, 14.060717, 17.11837, 14.533429, 14.386257, 14.179025, 17.853548, 18.51861, 15.594855, 10.636785, 1
7.18201, 15.80161, 12.021575, 18.208006, 14.268683, 12.592975, 13.598133, 14.539208, 18.178251, 15.4728565, 1
6.115683, 13.667821, 20.258705, 15.081872, 8.752866, 13.962954, 17.455177, 13.717741, 12.69615, 16.431238, 1
6.708334, 17.600193, 14.024885, 13.358778, 15.991451, 14.274955, 15.106745, 15.309292, 15.03889, 14.976426, 1
7.028145, 11.58839, 17.15154, 13.302226, 14.488475, 18.17802, 21.984669, 11.980446, 10.159707, 22.115456, 13.
318937, 8.2628765, 17.532877, 13.691584, 13.664025, 15.133626, 17.237192, 12.176755, 14.657728, 13.727439, 1
7.22878, 17.993723, 13.136601, 13.675345, 10.172514, 16.762394, 19.008375, 19.417763, 10.61526, 14.995265, 1
5.018923, 15.929445, 13.817588, 18.610064, 12.348105, 13.759414, 14.684467, 10.613819, 21.026968, 14.537818,
11.511883, 20.063242, 10.62088, 16.204227, 19.052652, 17.995594, 9.611455, 17.966152, 12.983304, 16.888102, 1
5.134622, 17.62019, 14.281899, 12.408835, 17.506752, 14.410634, 12.380137, 17.975138, 12.195547, 14.600738, 1
7.64347, 11.322446, 16.239613, 16.113054, 14.21729, 14.428794, 16.078081, 15.26083, 13.053129, 15.322625, 10.
403339, 20.29473, 17.200457, 15.243029, 11.604073, 9.306104, 15.702747, 21.209452, 12.865431, 19.944254, 11.1
37532, 16.196512, 12.678639, 16.24779, 17.852211, 10.3672285, 17.005352, 15.81584, 17.327547, 10.662447, 12.3
51477, 16.290592, 16.489883, 9.439708, 17.542095, 18.646524, 16.039326, 12.652621, 16.36947, 13.543517, 13.63
1627, 18.337942, 15.460545, 10.2680435, 20.211239, 12.175045, 18.61018, 13.6083355, 17.699532, 11.234109, 16.
019445, 15.174066, 14.851383, 14.65651, 14.667033, 11.540543, 19.512173, 14.129803, 16.703796, 13.564225, 10.
441517, 18.999105, 15.310347, 19.099813, 11.9661665, 13.274362, 17.803524, 13.438834, 13.061301, 12.586422, 1
9.180176, 12.361376, 13.62843, 13.834001, 17.91368, 12.666924, 16.736435, 15.321418, 17.297567, 13.072798, 1
4.189596, 13.911819, 14.145203, 17.156057, 16.458326, 12.266294, 16.22826, 13.941934, 18.646492, 11.259042, 1
7.060144, 13.399685, 13.980641, 14.279994, 12.370867, 18.864939, 20.155823, 9.452743, 14.990102, 13.900097, 1
5.500285, 15.343698, 17.776464, 11.699627, 15.230102, 12.575498, 19.431648, 11.960092, 16.777292, 12.2069645,
15.975023, 13.073662, 14.560835, 17.579859, 14.937737, 13.347309, 16.837744, 13.363096, 17.156515, 13.868135,

```
15.073072, 16.3917, 12.646945, 11.724125, 16.643673, 16.614332, 17.16364, 14.320042, 12.644684, 10.880729, 1
8.980312, 14.716359, 18.258339, 13.805546, 11.826669, 16.752192, 15.925724, 11.088282, 9.38462, 19.695421, 1
5.682587, 14.28243, 13.888289, 16.816998, 14.735312, 14.9795885, 14.875532, 15.185008, 17.519844, 10.94819, 1
1.742957, 13.523357, 20.6878, 17.758888, 11.38347, 15.584435, 15.188646, 8.949619, 22.064596, 13.743838, 15.8
37605, 14.982583, 18.622227, 13.1730175, 12.140763, 10.002997, 19.69942, 14.799428, 10.349631, 13.551051, 22.
327057, 12.6044235, 20.412529, 10.543749, 16.611856, 14.888528, 12.4988575, 11.945509, 10.847438, 23.795536,
13.251787, 14.036121, 17.94432, 14.725893, 13.723671, 16.48667, 17.857004, 13.035096, 13.288312, 14.582555, 1
3.139894, 17.366457, 13.290226, 17.950785, 12.87677, 14.47093, 14.558607, 15.710672, 13.426054, 18.45314, 12.
041872, 17.454254, 13.637524, 13.017083, 13.899079, 13.077382, 18.365902, 17.318281, 16.494629, 9.489585, 11.
994153, 13.502528, 20.216908, 12.236523, 15.3223, 17.611311, 14.072425, 15.147373, 15.449086, 18.823315, 11.8
96992, 13.495112, 15.888554, 14.2777405, 14.191615, 16.823137, 15.374583, 11.607042, 16.133387, 18.022013, 9.
022622, 15.270257, 14.082186, 15.342261, 15.374507, 15.95637, 12.684629, 14.34572, 13.636276, 17.098, 12.9510
69, 13.094991, 19.506292, 10.86543, 14.986725, 19.778408, 16.630123, 15.5059595, 11.68374, 15.150072, 18.9413
66, 9.07514, 12.822602, 13.2912035, 19.41474, 15.303057, 15.940334, 12.789762, 12.664404, 16.862495, 14.96250
7, 10.485828, 21.885115, 11.319109, 19.39128, 13.540423, 10.592427, 11.657458, 20.840954, 11.158973, 16.86558
7, 11.086812, 17.039907, 15.577641, 15.814671, 12.597066, 12.127156, 15.583228, 17.303196, 19.337677, 10.1735
36, 15.067557, 13.452419, 14.282837, 17.301044, 12.310091, 15.887098, 16.734713, 12.853629, 14.460432, 17.852
49, 11.55013, 16.109758, 17.456276, 14.118683, 14.965661, 15.569902, 18.018167, 9.861611, 17.08895, 10.82450
5, 18.359049, 15.425638, 14.6539, 17.272757, 11.905551, 12.390644, 13.011364, 20.33961, 15.296884, 9.729692,
20.786764, 12.970827, 15.543228, 16.262133, 16.983955, 13.245693, 14.03634, 13.058468, 13.667183, 18.62407, 1
4.21442, 17.585524, 12.040375, 18.426182, 12.814175, 12.759201, 10.781695, 16.63317, 17.630753, 16.902134, 1
6.528913, 9.961836, 15.86191, 15.616941, 12.438782, 14.162083, 16.260893, 13.806395, 13.875303, 17.292952, 1
2.834154, 16.668627, 15.063917, 12.126223, 16.019112, 17.40837, 9.999533, 11.982825, 16.824986, 15.890276, 1
6.903925, 16.121761, 10.4391, 10.246574, 10.7434225, 25.944004, 11.594532, 12.9657955, 21.354021, 15.328792,
19.021328, 8.691899, 16.256866, 11.862894, 16.772692, 13.606391, 17.257664, 13.248424, 15.479087, 16.440208,
11.797334, 11.028393, 22.840158, 9.361769, 13.569936, 11.667916, 20.467962, 11.222367, 21.019321, 11.446483,
9.83999, 20.298296, 14.122628, 12.516576, 15.035359, 17.489567, 16.702583, 14.602794, 12.648368, 15.371645, 1
4.486668, 14.506941, 14.773211, 15.817732, 13.554406, 13.875298, 16.209068, 14.224625, 11.623298, 16.342306,
17.020542, 14.878599, 15.96816, 13.192212, 13.592032, 11.040644, 21.187748, 16.641933, 15.825276, 11.147018,
19.310421, 8.711383, 16.818323, 12.6114, 17.146746, 14.626343, 13.535599, 14.090876, 17.355314, 9.096866, 19.
54184, 16.050781, 12.590526, 17.104527, 14.696596, 13.273609, 16.195498, 14.974576, 13.367667, 17.27247, 13.4
71331, 15.981864, 14.348134, 13.9015665, 13.9211445, 16.881691, 13.318841, 11.9325695, 14.975717, 18.384413,
16.56019, 14.035546, 13.560633, 12.746343, 15.529704, 16.6179, 12.501066, 16.151161, 16.023392, 18.402514, 1
2.637285, 12.992968, 13.461044, 16.029745, 14.9016485, 18.627163, 10.253714, 15.773221, 18.060158, 11.403034,
15.003764, 15.253163, 14.53421, 14.601961, 14.847423, 15.881792, 13.382047, 11.778285, 20.866465, 10.899697,
10.128246, 19.74259, 14.496617, 17.755966, 12.346558, 14.106708, 14.452901, 15.887939, 13.867091, 10.851022,
13.389172, 21.677809, 16.01243, 13.333299, 15.1813965, 16.227211, 12.415573, 16.056604, 13.65704, 12.088451,
19.801952, 11.124039, 17.438354, 16.199123, 12.705905, 13.98764, 18.561325, 17.342628, 15.662228, 10.395457,
14.605824, 16.19076, 13.267534, 13.6085825, 14.726711, 16.50448, 17.089905, 14.495174, 12.196352, 19.442923,
13.4026575, 10.592288, 19.151535, 11.254253, 13.759111, 13.437119, 19.167593, 10.905081, 15.871578, 10.94165
3, 18.32134, 16.737122, 13.972708, 13.361818, 18.72239, 13.595515, 11.288157, 16.639698, 12.246157, 15.69879
3, 19.314459, 13.601811, 10.493315, 13.925798, 15.235438, 15.413103, 18.427906, 14.413361, 10.605332, 13.2953
3, 14.40527, 17.217083, 12.863998, 18.552454, 12.403831, 14.323955, 13.303185, 17.362484, 15.881027, 11.19896
```

```
2, 17.998201, 12.631641, 16.104296, 15.921171, 11.160052, 21.108131, 11.340922, 16.029617, 14.221658, 13.9073
74, 17.364723, 13.2664995, 13.43753, 17.124184, 12.881078, 14.227555, 11.99224, 17.054867, 15.536351, 14.0122
88, 17.946, 11.75754, 15.22968, 15.345866, 13.55671, 14.421916, 13.994825, 16.230042, 14.967597, 15.934248, 1
3.024323, 16.718603, 13.356892, 14.135123, 17.472155, 11.372467, 15.678803, 11.786448, 19.569567, 12.541678,
10.039882, 18.391083, 16.33818, 10.519318, 18.463036, 15.571043, 18.062815, 11.709621, 14.516229, 14.6492, 1
2.790061, 17.479315, 17.529228, 13.909105, 12.359572, 15.360757, 13.974155, 15.164972, 19.063108, 13.067506,
11.454042, 15.790584, 14.2080345, 14.241206, 12.949148, 19.273403, 11.363548, 13.096083, 16.215954, 15.13435
6, 16.840937, 13.324507, 13.975912, 18.48975, 10.939835, 14.920654, 15.560084, 14.69669, 13.92288, 18.435938,
14.341268, 10.663308, 14.173527, 14.346401, 16.215185, 15.355883, 14.39345, 14.525118, 12.030346, 14.820877,
18.27353, 15.068222, 11.087673, 19.146616, 11.742936, 11.4708185, 22.958181, 8.884927, 19.8404, 15.834254, 1
2.5077095, 15.227031, 17.223671, 16.447989, 11.231082, 17.194471, 12.026808, 11.418607, 22.67523, 14.714695,
14.589067, 15.183433, 12.265306, 16.890472, 15.247968, 12.797041, 18.28026, 12.826608, 16.976336, 10.48911, 1
7.473524, 18.419079, 15.663256, 8.926635, 11.185191, 21.168188, 11.207945, 14.532735, 14.412255, 15.597155, 1
6.931536, 15.7576885, 10.762794, 12.108013, 13.830868, 19.480162, 14.3838, 17.18651, 12.169059, 15.806444, 1
3.42717, 15.199166, 15.623271, 14.147653, 14.540621, 13.091417, 14.189287, 17.667727, 13.124523, 12.273387, 2
0.113495, 21.094303, 10.581863, 11.990229, 14.050545, 14.621372, 15.9007635, 11.441918, 13.45004, 20.802216,
13.140614, 17.01164, 14.003534, 14.32954, 12.779352, 18.024796, 14.798238, 14.50967, 15.104058, 15.995246, 1
4.206417, 13.900063, 16.28074, 13.235803, 14.757269, 14.127949, 17.02199, 12.764544, 13.952452, 17.25138, 12.
636151, 18.904226, 12.551273, 12.247633, 20.581911, 10.753155, 12.4625845, 20.565216, 9.375124, 14.163561, 1
4.179993, 16.720207, 13.019394, 12.585578, 13.896925, 18.821112, 16.812332, 15.1112175, 11.690505, 16.294586,
12.481825, 15.785943, 13.3797035, 19.617004, 10.331656, 16.317078, 14.166406, 13.554507, 17.26715, 15.513424,
10.530599, 14.784344, 14.62112, 14.926216, 14.91532, 15.562878, 13.5718, 13.239931, 16.100565, 15.056364, 12.
808187, 10.384441, 22.943436, 13.662134, 15.121999, 15.722499, 20.095755, 9.881689, 14.163315, 13.678707, 18.
213356, 11.7256365, 16.573069, 13.084421, 14.573074, 12.2828865, 17.706093, 14.159026, 20.908026, 11.928689,
10.514077, 18.000292, 13.160521, 12.651201, 15.774292, 11.68258, 17.409048, 18.625841, 17.2179, 6.6165, 17.66
5318, 13.199426, 13.05477, 15.632527, 12.100215, 17.040705, 19.387783, 12.964316, 11.080627, 12.60076, 16.128
239, 15.8524, 13.155019, 14.136761, 17.69607, 18.168339, 10.773457, 15.489816, 17.209154, 10.306928, 17.38585
3, 14.8382225, 17.106728, 11.652118, 13.342222, 18.941202, 11.186943, 14.097118, 12.808908, 18.126997, 13.413
674, 15.991845, 14.903968, 14.902415, 10.720599, 19.763273, 13.833645, 17.169764, 12.818871, 19.86112, 9.5132
475, 14.906755, 8.05518, 22.700861, 13.173796, 16.98652, 11.842124, 15.698553, 20.557632, 10.3633375, 12.9338
55, 14.635949, 19.024776, 9.420798, 12.773104, 14.679515, 17.426075, 13.568802, 17.834564, 12.346049, 11.6590
98, 16.425938, 16.675564, 12.73128, 18.167305, 13.01002, 15.106964, 16.849648, 11.593138, 19.297195, 11.04198
3, 13.67522, 17.636068, 14.303994, 11.610624, 15.297789, 12.352098, 17.167439, 9.621185, 11.333369, 25.7868,
14.042336, 14.071915, 16.554367, 14.200883, 14.153409, 16.210281, 15.740045, 13.56933, 15.029915, 14.667629,
13.740406, 16.119497, 11.852841, 15.602388, 17.36914, 16.949821, 12.868908, 14.383522, 12.698724, 16.170017,
15.560237, 15.488454, 12.837579, 16.32016, 14.849874, 14.418586, 15.090319, 14.038269, 14.699589, 15.732924,
14.352166, 18.063358, 10.972548, 13.769817, 19.907877, 9.386812, 10.440122, 18.595163, 15.366041, 12.243412,
20.878439, 10.096968, 15.799201, 11.681986, 17.366364, 14.883111, 15.783844, 13.326098, 18.780582, 10.857383,
14.54619, 18.08442, 12.046904, 13.920029, 13.859361, 13.07095, 18.09402, 16.844551, 12.4244995, 15.078699, 1
4.549092, 12.04662, 18.437984, 19.319237, 10.530347, 14.352223, 18.779545, 9.973565, 15.635339, 13.983281, 1
7.005518, 12.780511, 16.239342, 15.245922, 12.220216, 18.105425, 13.962709, 11.450478, 17.001703, 13.177448,
13.880629, 15.8907385, 16.242775, 11.337335, 16.126844, 16.233435, 11.040797, 14.637736, 8.35775, 23.101912,
19.250393, 11.627392, 12.959756, 11.088155, 17.958687, 15.340422, 16.245346, 12.924108, 15.191539, 15.756866,
```

10.46704, 18.949984, 13.623505, 12.226031, 19.449657, 13.569476, 17.18685, 13.140439, 13.426106, 10.787092, 2
1.488214, 15.812302, 16.977463, 10.490099, 19.969316, 12.527186, 10.848353, 12.406582, 11.776845, 21.63258, 1
3.862865, 11.161043, 20.420609, 14.134306, 14.663262, 15.598956, 19.65986, 12.387099, 11.397453, 12.264501, 1
5.540844, 16.865788, 15.255225, 16.058437, 12.463434, 11.509426, 13.553899, 20.452698, 14.745982, 13.217952,
16.685919, 15.79936, 13.337021, 15.222295, 19.41845, 11.54892, 12.820243, 14.078289, 12.07568, 19.009342, 14.
725328, 14.754379, 14.715716, 12.055114, 14.469615, 18.631453, 14.782427, 17.699259, 10.848201, 11.953624, 1
7.66395, 14.609313, 13.8860855, 14.092579, 16.718784, 15.239465, 13.435683, 15.772546, 12.249868, 16.84602, 1
5.2386, 14.789117, 10.597402, 19.979902, 15.510631, 14.890383, 13.619573, 16.3041, 16.417315, 10.516416, 14.2
21878, 16.665247, 12.8701935, 17.484581, 10.915055, 16.16537, 16.088991, 13.817424, 14.164293, 12.790746, 17.
736547, 13.366501, 16.009602, 13.983158, 14.0274515, 13.587808, 16.700808, 13.704499, 19.369368, 12.884459, 1
1.196174, 13.902688, 17.376661, 12.408574, 12.761568, 16.973965, 14.411288, 14.187277, 15.819973, 13.9905, 1
4.86654, 14.492018, 14.872012, 15.278436, 16.301157, 11.952077, 13.59562, 14.605831, 16.37435, 14.801832, 15.
637017, 13.495797, 14.597858, 12.023028, 18.45772, 13.853804, 14.124651, 16.625187, 14.970829, 9.608948, 21.0
1178, 14.72061, 16.434515, 12.6152935, 16.770779, 13.743419, 13.347059, 16.214937, 16.838215, 10.086276, 19.9
09668, 12.055887, 11.47989, 11.801802, 12.279219, 21.702732, 14.886012, 14.980686, 14.176658, 16.124767, 12.9
953985, 15.193559, 17.637234, 15.161211, 10.401243, 11.062389, 13.199299, 21.43802, 12.38741, 14.427141, 18.1
6056, 16.364204, 12.841416, 14.987599, 13.021401, 17.742832, 12.98647, 17.384577, 15.080214, 10.809603, 14.22
0043, 12.218945, 18.6411, 11.673451, 16.11113, 16.903605, 16.785854, 13.685329, 13.434877, 12.487662, 18.9265
67, 12.271395, 15.516072, 11.818554, 17.402004, 12.819803, 14.086971, 18.045557, 14.304906, 16.866856, 12.508
779, 17.79416, 12.915964, 13.07224, 13.108784, 14.454413, 17.167212, 14.617123, 12.45937, 17.837292, 15.48277
95, 12.591088, 16.497356, 14.714186, 15.244269, 14.085467, 11.647915, 15.6446705, 17.488546, 16.49253, 13.722
085, 13.74138, 17.006645, 15.806168, 10.390993, 15.151352, 10.835792, 19.155409, 16.43507, 11.961368, 16.0761
15, 14.101331, 11.26034, 19.992054, 12.48541, 20.211077, 10.516765, 12.454114, 13.880709, 18.716707, 14.23971
75, 15.277052, 14.651465, 13.328482, 20.774073, 8.692376, 19.703575, 11.788075, 12.046733, 13.474126, 17.8398
88, 12.318022, 16.690813, 12.2665415, 15.34756, 11.763482, 12.970139, 20.736338, 15.311225, 15.265401, 13.207
377, 15.621365, 17.570366, 9.844325, 14.579054, 11.845499, 18.689192, 15.305401, 13.093414, 16.099367, 15.206
465, 12.213471, 17.281712, 15.311456, 15.241008, 13.251828, 14.545485, 17.573782, 11.277582, 14.662801, 13.26
0154, 16.682531, 19.150278, 13.952419, 9.956566, 13.382535, 13.413208, 18.111628, 14.280745, 13.899515, 16.32
9279, 16.772331, 15.398434, 11.139138, 15.89447, 15.914256, 11.639525, 13.740158, 13.991305, 16.96044, 11.862
617, 15.794316, 16.953432, 15.8104315, 15.971568, 11.67029, 12.796819, 14.066474, 18.05768, 15.079949, 10.583
561, 19.55614, 11.65814, 13.599496, 20.144276, 19.110723, 15.039863, 8.605272, 18.00454, 14.217675, 11.15424
3, 11.895845, 17.707943, 14.465552, 16.925535, 14.382182, 12.294214, 15.484604, 14.167324, 14.4255295, 13.701
847, 15.628638, 14.901854, 13.196085, 13.437114, 18.307064, 13.155884, 18.805887, 11.392524, 18.378273, 12.27
0859, 13.056453, 17.590511, 11.366887, 15.327891, 13.39903, 16.983921, 13.417323, 16.060701, 13.662136, 14.30
1573, 12.277385, 15.855397, 16.354067, 16.29285, 14.383865, 13.044362, 13.228906, 20.982079, 8.49654, 15.7619
67, 12.782658, 15.840661, 16.171211, 14.697844, 12.75871, 12.872917, 12.2786255, 20.189728, 10.970502, 14.421
441, 19.86758, 17.867842, 9.182957, 17.761448, 16.04075, 15.401873, 12.104501, 16.730246, 11.250767, 16.54999
7, 15.655631, 12.583101, 16.209898, 16.537622, 12.789132, 14.80464, 17.203737, 17.130474, 8.37071, 14.850403,
17.753428, 10.585364, 11.1840315, 17.765242, 15.303511, 15.381916, 12.814933, 16.21266, 16.220406, 15.703574,
11.494238, 16.74708, 12.288842, 15.17661, 11.528927, 16.811712, 16.063414, 18.012856, 12.971427, 12.678276, 1
5.182011, 14.324308, 14.53048, 9.580349, 16.260118, 19.315258, 15.141111, 15.689036, 12.847184, 17.057863, 1
6.457624, 9.337285, 12.616623, 17.827938, 13.343282, 12.441564, 16.918873, 14.768733, 18.50444, 13.42001, 11.
4438, 13.234211, 15.846483, 15.078539, 15.829091, 15.375892, 12.363623, 13.655703, 16.222738, 14.132896, 13.7

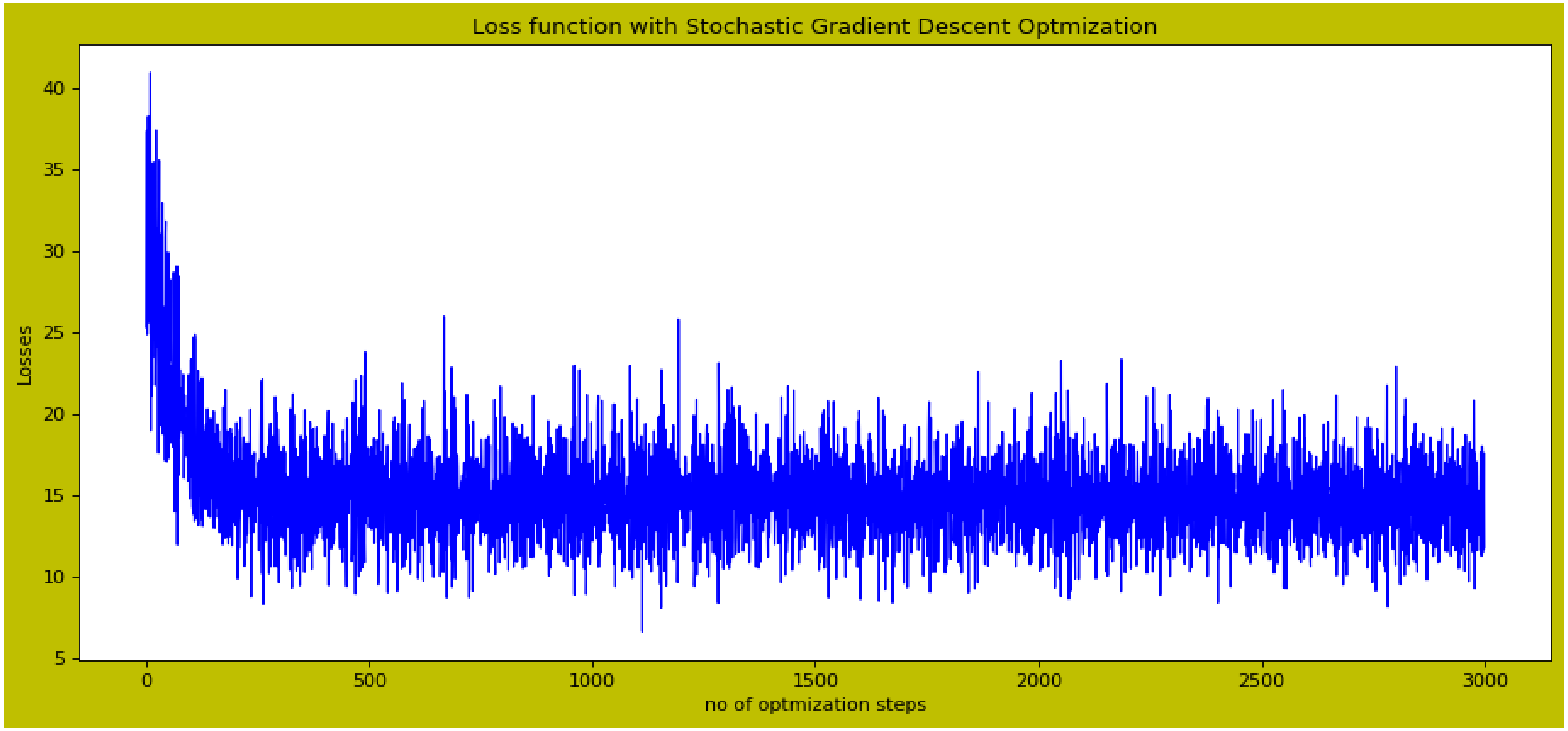
```
26344, 16.726034, 13.33402, 17.112892, 12.956411, 13.744726, 11.626173, 19.186945, 12.929231, 12.358207, 16.4
58733, 15.423722, 16.62368, 13.12881, 14.196218, 16.017525, 17.02575, 10.043223, 14.010996, 17.15203, 12.4213
17, 15.272366, 14.72958, 13.90908, 15.648065, 16.4132, 11.282259, 13.418591, 12.582265, 19.013376, 20.70499,
13.0465765, 9.044904, 13.1492195, 19.740282, 10.22112, 15.55608, 13.659774, 14.868685, 16.197802, 13.611251,
14.198863, 16.302889, 10.254693, 18.346272, 16.997576, 12.977261, 13.900381, 13.423288, 18.911251, 10.902861,
13.522167, 15.284351, 15.493032, 16.44498, 13.031801, 14.56459, 14.305651, 12.950758, 17.416468, 13.621576, 1
4.4936695, 16.39681, 17.29138, 12.470929, 14.229797, 10.216446, 17.375145, 16.964237, 10.694279, 18.742977, 1
4.606803, 14.001443, 14.464868, 15.855491, 15.220754, 15.307105, 13.236771, 11.078637, 15.88759, 17.807774, 1
4.267985, 15.696784, 13.916885, 18.257341, 12.158115, 13.392125, 13.589664, 17.87959, 12.025443, 11.905461, 1
6.140158, 16.416842, 19.279633, 13.05133, 10.868437, 17.832119, 11.811298, 14.313616, 14.059909, 18.438574, 1
0.657806, 15.412184, 10.210414, 19.533, 13.44858, 14.93987, 15.992986, 14.79934, 16.2664, 12.547463, 16.74638
6, 13.964556, 12.950623, 11.522473, 16.141771, 16.927826, 17.804052, 9.01276, 17.954876, 12.519438, 15.29773
3, 16.643795, 15.179053, 16.706163, 11.463216, 13.3658495, 15.255663, 15.607977, 15.423277, 11.904831, 17.281
775, 9.257799, 19.122114, 16.011374, 16.589943, 13.241497, 14.059791, 13.647062, 9.597931, 22.552807, 12.7083
57, 16.418852, 15.018816, 14.115509, 16.245155, 13.391565, 14.735683, 12.565054, 17.354816, 15.331704, 14.403
575, 14.171161, 15.122548, 17.28942, 10.74097, 17.318373, 13.2599535, 13.101255, 14.690635, 16.272604, 12.676
455, 10.664394, 20.713963, 12.131899, 12.349803, 16.449715, 15.390365, 16.50234, 12.551891, 15.110214, 12.867
31, 17.384804, 13.513028, 11.534474, 17.216227, 15.492696, 14.620045, 16.249548, 12.73793, 17.549772, 11.3849
18, 15.160128, 13.3491125, 13.452165, 18.038275, 14.79175, 14.213524, 15.185845, 18.005909, 13.488954, 11.905
66, 12.845565, 13.255595, 18.887615, 15.729516, 15.657067, 12.083762, 12.512702, 16.044239, 15.712909, 14.545
451, 14.375074, 15.295855, 16.86346, 14.056816, 12.637719, 16.674328, 11.529528, 16.187674, 18.121683, 13.087
486, 12.283107, 18.03738, 13.791294, 11.488091, 14.056618, 13.488632, 17.034746, 14.403902, 13.395092, 16.680
374, 20.310553, 12.199791, 10.615044, 15.234392, 13.467239, 15.538409, 15.149117, 11.440884, 18.262012, 12.75
8684, 16.935062, 14.28816, 12.937381, 15.722172, 15.600039, 14.305973, 17.143711, 11.965411, 16.447716, 15.07
125, 11.846072, 14.3860035, 17.52658, 11.341344, 13.370311, 15.70853, 15.032487, 11.503632, 14.520576, 18.922
367, 12.436576, 13.630914, 18.974466, 14.698238, 10.73188, 19.808195, 13.387948, 10.816686, 21.296816, 15.400
515, 11.975468, 17.244595, 12.541524, 14.81248, 17.256975, 13.366774, 14.237706, 16.892681, 13.025748, 16.375
587, 14.626319, 14.683827, 12.951905, 16.927599, 15.143046, 17.528679, 10.348106, 14.00749, 15.711192, 14.226
44, 17.594654, 14.965997, 10.525165, 15.921757, 13.837875, 14.130534, 13.225449, 12.611793, 19.226542, 12.733
438, 15.722612, 15.860332, 13.569208, 14.997562, 15.624183, 10.6120405, 15.2435, 19.178663, 12.446155, 12.724
916, 20.06035, 11.811615, 15.574009, 17.188158, 13.574831, 14.062766, 16.847021, 14.884297, 11.404188, 18.576
59, 9.304296, 14.9216175, 21.284273, 16.183508, 15.77596, 11.259351, 12.014408, 18.798706, 12.772376, 18.3754
44, 9.893553, 16.036606, 13.918786, 8.787709, 23.249115, 13.2319975, 17.317812, 13.083435, 11.489175, 14.0989
34, 19.519888, 13.710816, 17.51865, 12.203483, 16.357704, 13.185447, 14.440042, 11.258133, 12.866849, 21.4324
86, 18.069908, 8.635624, 18.031591, 15.163389, 18.489468, 9.109262, 15.436628, 14.898061, 13.385834, 13.85912
2, 12.625844, 18.389896, 16.287437, 15.532379, 11.527481, 9.8343525, 16.299456, 18.82438, 12.756315, 17.66173
6, 13.27308, 15.669422, 17.394535, 9.810288, 14.475357, 13.773375, 16.072714, 13.92447, 14.84661, 15.395983,
18.021982, 12.864413, 12.687498, 11.378515, 19.708347, 12.404624, 17.143862, 14.06417, 12.309382, 15.686232,
13.688447, 14.642829, 14.252112, 16.321482, 13.009058, 18.241491, 13.641124, 11.2502575, 14.709605, 13.43556
6, 16.184387, 13.687391, 17.690575, 12.008196, 14.599897, 15.625168, 13.460002, 12.249639, 13.910954, 18.7662
83, 17.361961, 15.66547, 9.844615, 13.692727, 17.026506, 12.889806, 17.95843, 12.718564, 12.868142, 16.35729
6, 14.162406, 13.0813055, 13.570494, 15.705763, 14.706933, 13.5281725, 14.127572, 16.823978, 13.81465, 14.238
957, 16.314219, 15.5962, 13.413788, 15.0828705, 13.414773, 10.304203, 21.825937, 18.399601, 10.036146, 15.772
```

036, 15.269428, 15.508137, 12.782281, 14.4621935, 12.440489, 17.774557, 15.004986, 11.698949, 17.99315, 10.93
6354, 17.433144, 15.834268, 12.35037, 14.100401, 18.388498, 15.125822, 14.703625, 14.005873, 12.294361, 14.73
366, 17.593306, 12.748585, 16.67237, 14.487178, 18.412838, 11.954751, 13.313863, 13.489832, 9.072536, 23.3637
33, 16.940264, 11.331113, 15.938194, 15.552392, 12.483803, 16.258236, 18.061121, 10.235926, 15.8968115, 13.35
3372, 16.082964, 14.4574585, 15.155975, 12.700302, 16.544611, 14.435318, 16.129686, 13.045222, 18.137306, 14.
379715, 10.478484, 14.623082, 15.525657, 13.579091, 13.325393, 13.35104, 18.027576, 13.596929, 14.811689, 15.
779268, 14.98408, 14.918293, 13.899083, 10.672248, 16.297825, 17.66545, 15.683055, 15.426307, 12.300735, 15.6
81401, 14.117517, 13.98669, 14.946645, 16.655704, 11.726484, 14.072376, 13.248318, 17.17584, 18.252462, 12.37
3079, 12.951136, 13.530975, 18.010477, 11.733294, 11.6994095, 21.044882, 10.191384, 13.535801, 15.936338, 14.
431215, 16.217428, 11.61415, 16.551744, 13.698011, 17.370878, 12.391948, 13.934716, 14.339641, 15.895401, 10.
304418, 21.604061, 11.24234, 12.288476, 19.377401, 11.619283, 12.1063795, 16.69138, 15.300029, 19.007965, 11.
694861, 12.863397, 15.602079, 13.285214, 15.195724, 15.637251, 18.154781, 8.874055, 13.182271, 12.513996, 19.
344381, 11.713989, 14.799229, 18.183325, 13.370092, 16.626112, 13.731067, 15.015682, 14.483741, 14.423782, 1
4.671149, 16.028898, 12.814688, 17.87863, 11.13324, 14.993093, 11.650032, 12.548046, 21.17771, 10.009299, 20.
197575, 13.423613, 16.73169, 12.425023, 14.752297, 16.380037, 15.652308, 11.123527, 14.236627, 17.765661, 11.
1679535, 14.519197, 15.060059, 14.236185, 16.741583, 12.516009, 14.601234, 14.079463, 15.207378, 14.639051, 1
7.25469, 14.545212, 11.43051, 17.721748, 13.361187, 12.326932, 14.957831, 13.667401, 15.457918, 14.058334, 1
8.206438, 10.756186, 14.964406, 12.308247, 17.203861, 15.245653, 15.176971, 13.187773, 15.514498, 13.514731,
14.956761, 13.874245, 18.83273, 10.304152, 18.449633, 13.972983, 10.5383215, 16.11414, 14.736623, 12.594317,
12.33539, 17.838982, 13.471428, 14.74086, 13.776225, 15.604828, 16.939762, 11.772413, 15.355721, 14.198944, 1
2.063763, 18.478092, 13.774887, 17.497362, 12.093358, 18.416407, 13.436532, 11.268504, 15.7651415, 11.743598,
16.915241, 17.001762, 11.7715645, 15.263803, 11.661015, 14.212397, 18.997335, 17.567314, 13.997689, 11.66734
6, 11.612884, 13.18372, 20.476976, 10.018727, 20.954578, 12.42191, 12.2023115, 14.6172285, 17.800228, 13.6806
99, 12.913218, 18.07385, 18.04188, 12.3901, 13.12954, 11.859781, 14.080886, 18.868835, 13.117682, 16.235714,
14.544625, 13.832341, 14.938209, 15.32545, 13.943263, 20.17249, 8.3684025, 19.782263, 11.157405, 12.45764, 1
3.293757, 14.621416, 16.432926, 12.044268, 19.23801, 12.030409, 17.705624, 13.427425, 12.201983, 11.998737, 1
7.962551, 13.779876, 13.607766, 14.780173, 15.711675, 18.171652, 11.8667345, 13.548331, 17.080437, 14.435043,
11.755696, 17.545122, 13.253994, 12.655197, 9.375957, 17.743752, 17.400576, 13.722526, 16.322834, 13.631175,
15.004802, 12.98901, 16.263428, 15.414198, 16.293398, 11.467595, 14.651851, 15.074184, 14.065316, 11.46485, 1
3.395649, 20.27923, 12.376482, 16.67373, 14.858919, 16.857618, 13.699528, 12.976893, 14.267116, 13.706504, 1
6.29141, 13.590832, 13.439473, 17.470312, 13.660989, 13.520023, 17.375479, 18.761251, 11.882111, 12.805269, 1
3.096745, 18.733242, 11.246603, 14.658793, 11.417141, 18.749546, 18.64097, 12.818346, 11.7739935, 10.788629,
17.51929, 15.812869, 12.3469, 12.56786, 20.2131, 11.927424, 16.757557, 15.419312, 14.155589, 15.030729, 14.72
9715, 11.457676, 19.563427, 12.373054, 15.60652, 13.41709, 14.898444, 12.955448, 15.613362, 15.523988, 16.000
261, 11.854282, 16.380583, 19.645515, 11.402464, 12.277072, 15.083211, 12.870223, 16.311945, 14.094736, 15.62
4788, 14.018947, 14.1537, 17.52985, 11.465361, 11.490755, 16.293993, 16.507061, 13.577675, 13.125351, 17.9613
46, 11.0424, 15.177302, 18.555117, 16.428179, 12.234362, 15.378782, 18.595095, 13.39614, 11.07994, 20.83901,
11.773937, 10.205231, 12.635383, 16.852282, 14.31685, 11.233876, 17.283028, 15.578319, 17.585714, 10.789089,
15.763369, 14.124604, 15.754244, 13.887686, 12.984734, 18.061172, 12.305568, 15.24265, 12.636093, 16.373869,
11.883879, 21.472372, 9.302201, 11.204059, 13.815386, 20.17389, 15.2536335, 18.171282, 9.258769, 16.670118, 1
3.591143, 13.329051, 12.906673, 16.6382, 14.232873, 15.885498, 15.44882, 11.87837, 15.317006, 13.539245, 15.1
21048, 14.65547, 15.071686, 14.013547, 18.495098, 12.112474, 12.793965, 14.408199, 16.456032, 12.486447, 13.2
26511, 18.332907, 11.621068, 12.185109, 15.107239, 17.088501, 14.143949, 11.06335, 19.765045, 17.421665, 14.7

06994, 10.856732, 13.376748, 16.189949, 14.21577, 12.358069, 14.318227, 17.890463, 11.108218, 19.940577, 12.239176, 18.429403, 11.888126, 13.186828, 13.223085, 15.472324, 15.245286, 17.008663, 11.869835, 15.009254, 17.648664, 14.389232, 10.962201, 10.6230545, 17.487968, 16.107843, 11.196993, 16.508036, 16.605417, 16.182987, 14.558465, 12.67111, 16.41851, 13.201369, 14.16938, 17.35161, 12.5511, 13.702986, 16.530888, 11.339628, 16.431871, 15.308146, 13.726972, 14.841918, 16.750593, 13.6753, 13.115038, 12.438365, 14.880886, 17.089216, 11.537884, 14.04845, 19.344145, 13.805035, 15.232703, 14.85023, 11.1592, 17.624037, 15.146471, 12.1267605, 13.290087, 19.496511, 15.284953, 15.457319, 12.6929865, 15.04175, 13.921929, 14.916619, 12.362676, 16.801548, 14.65818, 16.938293, 11.977593, 15.015427, 13.489622, 18.340248, 11.244337, 12.653948, 15.697432, 15.728073, 21.120575, 10.059139, 12.092553, 12.630147, 16.927597, 14.145501, 14.819102, 11.722805, 18.042383, 13.108146, 14.569554, 16.567425, 14.709617, 16.610273, 11.8612, 17.970856, 9.478779, 16.888775, 18.486532, 13.54104, 10.950555, 14.012434, 13.966732, 16.154045, 10.077029, 16.76362, 17.6751, 11.924892, 17.898247, 13.863215, 13.047232, 15.192875, 15.856853, 12.12714, 14.588713, 17.902172, 15.861917, 15.114672, 12.327014, 15.220123, 16.429832, 11.460714, 14.732775, 12.717715, 16.890173, 11.887333, 19.794142, 11.336363, 16.289354, 11.25008, 16.73145, 14.790229, 16.494211, 11.902638, 11.65296, 16.450773, 16.061651, 16.597021, 14.133338, 12.637336, 15.383251, 14.048079, 14.387658, 12.304723, 13.307364, 19.161505, 14.7708435, 12.636473, 16.927921, 14.681926, 10.522944, 19.7483, 15.629551, 13.460257, 14.752059, 13.090252, 12.618148, 19.161165, 14.775074, 12.852088, 16.647509, 9.615036, 17.160585, 17.72613, 15.207438, 15.365016, 12.831131, 16.064646, 17.389727, 9.09862, 12.4542265, 13.252736, 19.093924, 12.835331, 18.379784, 12.00512, 15.351351, 16.649023, 11.002289, 15.045626, 13.411289, 15.554696, 12.5855665, 13.831356, 18.184637, 18.095364, 14.091711, 10.692024, 16.229649, 16.323853, 10.308716, 15.289089, 14.19733, 14.207588, 21.727364, 12.473054, 8.116942, 13.773948, 13.988749, 16.514452, 14.284498, 13.206048, 16.783092, 13.160711, 14.998596, 15.9813595, 17.489902, 11.687596, 14.566096, 12.586855, 15.960161, 15.417801, 12.130932, 10.704381, 22.88777, 12.910087, 14.343085, 17.103388, 16.968855, 13.35811, 13.084089, 16.135172, 17.019253, 9.499728, 15.642985, 13.167319, 15.19307, 13.534442, 17.213945, 12.604265, 11.572294, 13.5443, 19.849453, 10.06832, 14.19939, 20.91609, 11.367867, 17.88211, 14.523227, 13.625729, 14.882684, 15.47716, 10.727903, 16.85768, 16.69791, 14.354496, 13.836445, 15.925332, 17.964743, 11.995586, 13.587624, 14.84236, 11.034121, 18.831486, 13.332709, 17.158842, 12.913681, 19.426247, 10.845463, 13.205747, 15.258818, 17.257055, 10.284938, 17.845303, 12.94364, 12.564023, 14.296066, 13.783415, 15.993176, 18.223309, 12.686538, 12.319962, 12.524297, 16.043121, 15.379987, 18.792805, 12.012913, 12.496614, 12.513265, 17.439865, 13.621706, 15.368918, 17.53383, 9.780236, 13.253621, 16.659615, 13.6477, 15.357773, 12.168125, 16.744577, 12.218129, 18.520962, 12.529489, 15.28183, 11.994151, 17.06951, 17.607183, 11.256666, 14.986161, 12.612728, 15.9830675, 15.3464155, 11.786272, 16.321774, 15.978553, 13.180461, 14.617637, 16.415365, 10.850474, 15.502494, 18.303942, 10.910112, 17.62067, 15.422751, 13.285816, 12.348535, 19.128723, 14.759471, 13.188189, 16.144526, 15.499632, 13.9553175, 14.247599, 14.484643, 17.03175, 11.542536, 14.512092, 11.914421, 18.1029, 11.514363, 15.933921, 16.858482, 15.518073, 14.258144, 13.773544, 14.433567, 17.294714, 11.275172, 13.718906, 13.704834, 16.82247, 19.134508, 11.910974, 12.204514, 17.112267, 13.024563, 13.342186, 11.578861, 15.969822, 16.696758, 15.705811, 11.644087, 16.925716, 17.812382, 14.57661, 10.488227, 16.804077, 14.002422, 12.50461, 16.567583, 15.106764, 11.386771, 12.373625, 14.145957, 17.953554, 11.861858, 17.71369, 14.02165, 10.33392, 15.677531, 18.686937, 16.133352, 12.945272, 14.647673, 14.766483, 15.571483, 13.045312, 9.683633, 16.657278, 18.24878, 12.142872, 15.651537, 16.412827, 17.76826, 12.429321, 13.174803, 11.487698, 12.845669, 20.815304, 17.5097, 9.253632, 17.584833, 14.417603, 12.991782, 16.816158, 17.062263, 11.590359, 15.257426, 15.267408, 14.313651, 14.049413, 12.551433, 14.309808, 17.607244, 15.227532, 11.268445, 17.980982, 17.935534, 11.464559, 14.227869, 13.778329, 17.58535, 11.772151,

In [80]:
```
fig, ax = plt.subplots(num=None, figsize=(14, 6), dpi=80, facecolor='y', edgecolor='k')
size = len(lr.losses)
ax.plot(range(0,size), lr.losses, '-', color='blue', animated = True, linewidth=1)
plt.xlabel('no of optmization steps')
plt.ylabel('Losses')
plt.title('Loss function with Stochastic Gradient Descent Optmization')
```

Out[80]:
```
Text(0.5, 1.0, 'Loss function with Stochastic Gradient Descent Optmization')
```



We can see how model's loss is almost constant after 500th batch.

In [81]:
```
min(lr.losses)
```

Out[81]:
```
6.6165
```

Minimum Loss fucntion we achieved is 7.1 approximately and thus accuracy of 92.9%. But unfortunately this result is on training dataset not on test dataset.

In [82]:
```
accuracy=100-min(lr.losses)
accuracy
```

Out[82]: 93.38350009918213

In [0]:
```
y_pred_nn=score(X_test)
```

In [85]:
```
print('MSE:', metrics.mean_squared_error(y_test, y_pred_nn))
```

```
MSE: 22.83741110622265
```

Let's try to do the Best Subset selection

In [0]:
```
import itertools
import time
import statsmodels.api as sm
```

In [0]:
```
def processSubset(feature_set):
  # Fit model on feature_set and calculate MSE
  model = sm.OLS(y_train,X_train)
  regr = model.fit()
  MSE = ((y_test-regr.predict(X_test)) ** 2).mean()
  return {"model":regr, "MSE":MSE}
```

In [0]:
```
def getBest(k):
  tic = time.time()
  results = []
  for combo in itertools.combinations(X_train.columns, k):
    results.append(processSubset(combo))
  # Wrap everything up in a nice dataframe
  models = pd.DataFrame(results)

  # Choose the model with the minimum MSE
  best_model = models.loc[np.argmin(np.array(models['MSE']))]
  toc = time.time()
  print("Processed", models.shape[0], "models on", k, "predictors in", (toc-tic), "seconds.")

  # Return the best model, along with some other useful information about the model
  return best_model
```

In [92]:
```
models_best = pd.DataFrame(columns=["MSE", "model"])

tic = time.time()
for i in range(4):
    models_best.loc[i] = getBest(i)

toc = time.time()
print("Total elapsed time:", (toc-tic), "seconds.")
```

```
Processed 1 models on 0 predictors in 0.015178918838500977 seconds.
Processed 50 models on 1 predictors in 0.3023357391357422 seconds.
Processed 1225 models on 2 predictors in 7.206106901168823 seconds.
Processed 19600 models on 3 predictors in 117.38464117050171 seconds.
Total elapsed time: 125.45728993415833 seconds.
```

In [93]:
```
print(models_best.loc[1, "model"].summary())
```

```
                            OLS Regression Results
=======================================================================================
Dep. Variable:                      y   R-squared (uncentered):                   0.527
Model:                            OLS   Adj. R-squared (uncentered):              0.370
Method:                 Least Squares   F-statistic:                              3.347
Date:                Sun, 15 Dec 2019   Prob (F-statistic):                    7.00e-09
Time:                        00:59:16   Log-Likelihood:                         -547.20
No. Observations:                 200   AIC:                                      1194.
Df Residuals:                     150   BIC:                                      1359.
Df Model:                          50
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x01           -3.1188      5.203     -0.599      0.550     -13.400       7.163
x02           -0.1291      4.746     -0.027      0.978      -9.507       9.248
x03            7.1450      4.637      1.541      0.125      -2.017      16.307
x04            2.6646      5.096      0.523      0.602      -7.404      12.733
x05           -2.8229      5.043     -0.560      0.576     -12.787       7.141
x06            3.2452      4.827      0.672      0.502      -6.292      12.782
x07           -3.2488      5.110     -0.636      0.526     -13.346       6.848
x08            4.7795      4.944      0.967      0.335      -4.989      14.548
x09            0.0828      4.776      0.017      0.986      -9.353       9.519
x10           -4.0881      5.181     -0.789      0.431     -14.325       6.149
x11            0.7750      3.406      0.228      0.820      -5.955       7.505
x12            1.7651      3.356      0.526      0.600      -4.867       8.397
x13           -2.2081      3.323     -0.664      0.507      -8.774       4.358
x14           -1.8011      3.587     -0.502      0.616      -8.889       5.287
x15           -1.2930      3.804     -0.340      0.734      -8.809       6.222
x16           -2.5538      3.544     -0.721      0.472      -9.556       4.449
x17           -1.0052      3.736     -0.269      0.788      -8.386       6.376
x18           -2.2132      3.821     -0.579      0.563      -9.764       5.337
x19            2.5080      3.502      0.716      0.475      -4.412       9.428
x20           -1.3930      3.301     -0.422      0.674      -7.916       5.130
x21            3.5559      3.595      0.989      0.324      -3.547      10.658
x22           -0.2649      3.510     -0.075      0.940      -7.200       6.670
x23           -3.6777      3.639     -1.011      0.314     -10.868       3.512
x24            0.4457      3.471      0.128      0.898      -6.413       7.304
x25            5.3167      3.282      1.620      0.107      -1.168      11.802
x26            0.7936      3.565      0.223      0.824      -6.251       7.838
x27            4.8582      3.840      1.265      0.208      -2.729      12.445
x28           -1.2869      3.189     -0.404      0.687      -7.588       5.015
x29           -1.9092      3.131     -0.610      0.543      -8.095       4.277
```

```
x30            6.5661      3.781      1.737      0.084      -0.904      14.036
x31            0.0796      0.358      0.223      0.824      -0.627       0.786
x32           -0.2485      0.359     -0.692      0.490      -0.958       0.461
x33            0.2262      0.367      0.616      0.539      -0.500       0.952
x34            0.7586      0.310      2.449      0.015       0.147       1.371
x35            0.9340      0.354      2.639      0.009       0.235       1.633
x36            0.2610      0.353      0.740      0.461      -0.436       0.958
x37           -1.0088      0.366     -2.755      0.007      -1.732      -0.285
x38            0.9213      0.338      2.730      0.007       0.254       1.588
x39           -0.3646      0.352     -1.036      0.302      -1.060       0.331
x40           -0.1663      0.354     -0.470      0.639      -0.865       0.532
x41            0.2125      0.340      0.625      0.533      -0.459       0.884
x42           -0.2185      0.368     -0.593      0.554      -0.946       0.509
x43            0.0813      0.337      0.241      0.810      -0.585       0.748
x44           -0.0183      0.358     -0.051      0.959      -0.726       0.689
x45            0.2680      0.355      0.756      0.451      -0.433       0.968
x46            0.2159      0.339      0.637      0.525      -0.453       0.885
x47            0.0775      0.339      0.229      0.819      -0.592       0.747
x48            0.3978      0.356      1.117      0.266      -0.306       1.102
x49           -0.2881      0.358     -0.804      0.422      -0.996       0.420
x50           -0.2767      0.364     -0.759      0.449      -0.997       0.443
==============================================================================
Omnibus:                       23.387   Durbin-Watson:                   2.126
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               28.905
Skew:                           0.793   Prob(JB):                     5.29e-07
Kurtosis:                       3.975   Cond. No.                         51.0
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [94]:
```
models_best
```

Out[94]:

	MSE	model
0	24.526909	<statsmodels.regression.linear_model.Regressio...
1	24.526909	<statsmodels.regression.linear_model.Regressio...
2	24.526909	<statsmodels.regression.linear_model.Regressio...
3	24.526909	<statsmodels.regression.linear_model.Regressio...