Report -

To predict binding affinity for a drug

Model used ------ Linear regression

Linear regression is a common Statistical Data Analysis technique. It is used to determine the extent to which there is a linear relationship between a dependent variable and one or more independent variables. There are two types of linear regression, simple linear regression and multiple linear regression.

In simple linear regression a single independent variable is used to predict the value of a dependent variable. In multiple linear regression two or more independent variables are used to predict the value of a dependent variable.

We have seen here linear relationship between various variables by plotting graphs so we used linear regression , ridgecv ,extratreeregressor .

Molecule to vector conversion has helped me in improving the performance of the model that is MSE and MAE of the model.

MAE score: 1.8517 MSE score: 2.2923

Data preprocessing

In [0]:
```
!wget -c https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh
!chmod +x Miniconda3-latest-Linux-x86_64.sh
!time bash ./Miniconda3-latest-Linux-x86_64.sh -b -f -p /usr/local
!time conda install -q -y -c conda-forge rdkit
#taking github code for mol to vector conversion
!pip install git+https://github.com/samoturk/mol2vec;
```

In [0]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import sys
import os
sys.path.append('/usr/local/lib/python3.7/site-packages/')
```

In [0]:
```
import numpy as np
import pandas as pd

from rdkit import Chem
from rdkit.Chem import DataStructs
from rdkit.Chem import AllChem
from rdkit.Chem import RDConfig
from rdkit import rdBase
from rdkit.Chem.Draw import IPythonConsole
from rdkit.Chem import Descriptors

from sklearn.linear_model import RidgeCV
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import ExtraTreesRegressor, RandomForestRegressor

from mol2vec.features import mol2alt_sentence, mol2sentence, MolSentence, DfVec, sentences2v
ec
from gensim.models import word2vec
```

In [186]:
```
df2 = pd.read_csv('/gdrive/My Drive/Dataset/AICrowd/Covid/train.csv')
df2.head()
```

Out[186]:

	SMILES sequence	Binding Affinity
0	CCNC(C)C(NC)c1ccccc1	-18.0861
1	CONC(=O)c1cncnc1	-17.5783
2	CCNC1CCCN(Cc2ccsc2)C1	-20.3645
3	CC(NC(=O)CSCCN)c1ccccc1	-19.3144
4	CCC(CS)CN(C)c1ccccc1	-15.8451

In [0]:
```
df2['mol'] = df2['SMILES sequence'].apply(lambda x: Chem.MolFromSmiles(x))
df2['mol'] = df2['mol'].apply(lambda x: Chem.AddHs(x))
df2['num_of_atoms'] = df2['mol'].apply(lambda x: x.GetNumAtoms())
df2['num_of_heavy_atoms'] = df2['mol'].apply(lambda x: x.GetNumHeavyAtoms())
```

In [188]:
```
# First we need to settle the pattern.
c_patt = Chem.MolFromSmiles('C')

# Now let's implement GetSubstructMatches() method
print(df2['mol'][0].GetSubstructMatches(c_patt))
```

```
((0,), (1,), (3,), (4,), (5,), (7,), (8,), (9,), (10,), (11,), (12,), (13,))
```

In [0]:
```
#We're going to settle the function that searches patterns and use it for a list of most com
mon atoms only
def number_of_atoms(atom_list, df):
    for i in atom_list:
        df['num_of_{}_atoms'.format(i)] = df['mol'].apply(lambda x: len(x.GetSubstructMatche
s(Chem.MolFromSmiles(i))))

number_of_atoms(['C','O', 'N', 'Cl'], df2)
```

In [190]:
```
#Leave only features columns
train_df = df2.drop(columns=['SMILES sequence', 'Binding Affinity', 'mol'])
y = df2['Binding Affinity'].values

print(train_df.columns)

#Perform a train-test split. We'll use 10% of the data to evaluate the model while training
 on 90%

X_train, X_test, y_train, y_test = train_test_split(train_df, y, test_size=.1, random_state=
1)

test_df = df1.drop(columns=['SMILES sequence', 'Binding Affinity', 'mol'])
```

```
Index(['num_of_atoms', 'num_of_heavy_atoms', 'num_of_C_atoms',
       'num_of_O_atoms', 'num_of_N_atoms', 'num_of_Cl_atoms'],
      dtype='object')
```

In [0]:
```
def evaluation(model, X_test, y_test):
    prediction = model.predict(X_test)
    mae = mean_absolute_error(y_test, prediction)
    mse = mean_squared_error(y_test, prediction)

    plt.figure(figsize=(15, 10))
    plt.plot(prediction[:300], "red", label="prediction", linewidth=1.0)
    plt.plot(y_test[:300], 'green', label="actual", linewidth=1.0)
    plt.legend()
    plt.ylabel('logP')
    plt.title("MAE {}, MSE {}".format(round(mae, 4), round(mse, 4)))
    plt.show()

    print('MAE score:', round(mae, 4))
    print('MSE score:', round(mse,4))
```

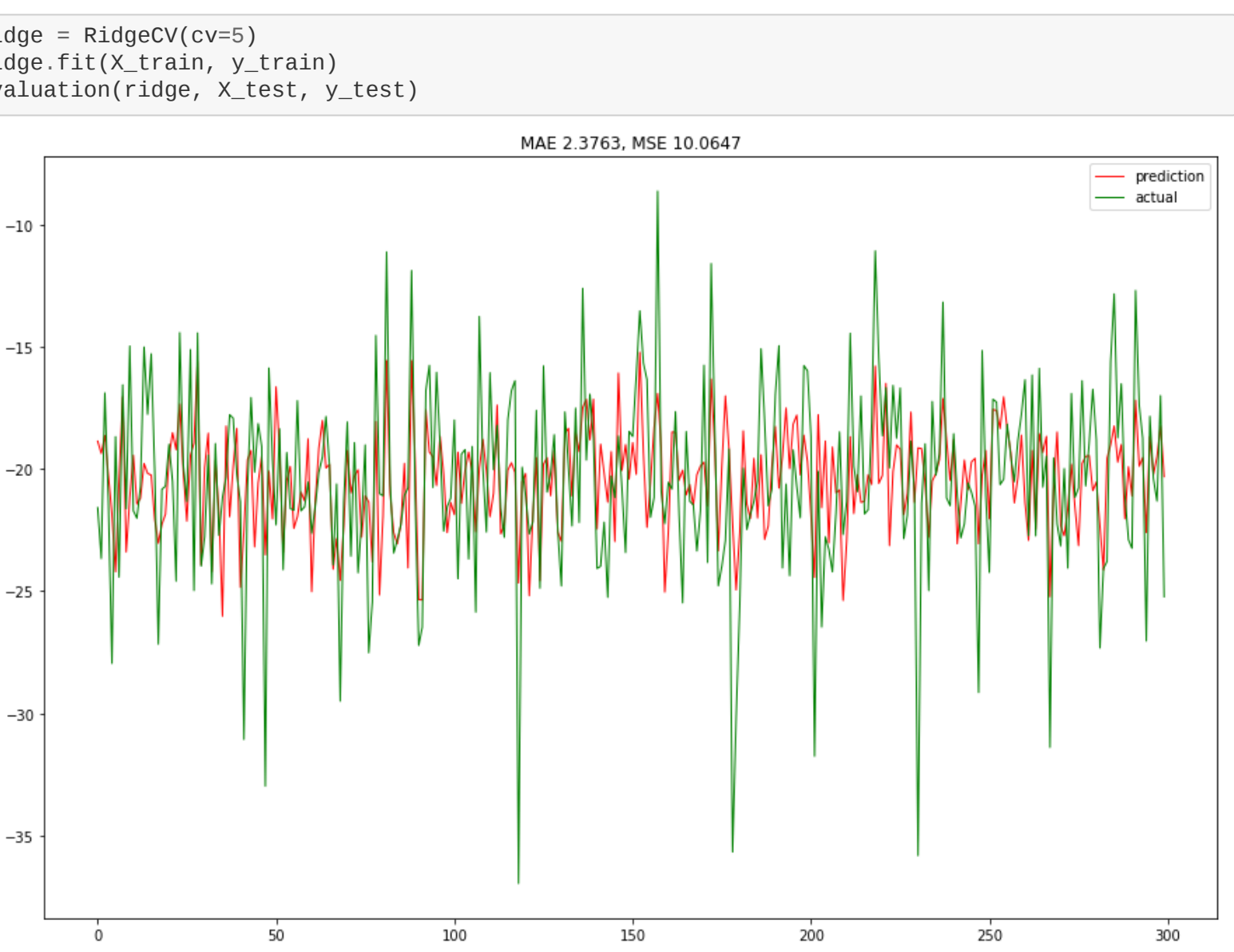
In [0]:
```
df2['tpsa'] = df2['mol'].apply(lambda x: Descriptors.TPSA(x))
df2['mol_w'] = df2['mol'].apply(lambda x: Descriptors.ExactMolWt(x))
df2['num_valence_electrons'] = df2['mol'].apply(lambda x: Descriptors.NumValenceElectrons(x
))
df2['num_heteroatoms'] = df2['mol'].apply(lambda x: Descriptors.NumHeteroatoms(x))
```

In [193]:
```
train_df = df2.drop(columns=['SMILES sequence', 'Binding Affinity', 'mol'])
y = df2['Binding Affinity'].values

print(train_df.columns)

X_train, X_test, y_train, y_test = train_test_split(train_df, y, test_size=.1, random_state=
1)
```

```
Index(['num_of_atoms', 'num_of_heavy_atoms', 'num_of_C_atoms',
       'num_of_O_atoms', 'num_of_N_atoms', 'num_of_Cl_atoms', 'tpsa', 'mol_w',
       'num_valence_electrons', 'num_heteroatoms'],
      dtype='object')
```

Ridge regression ------- MAE AND MSE calculation

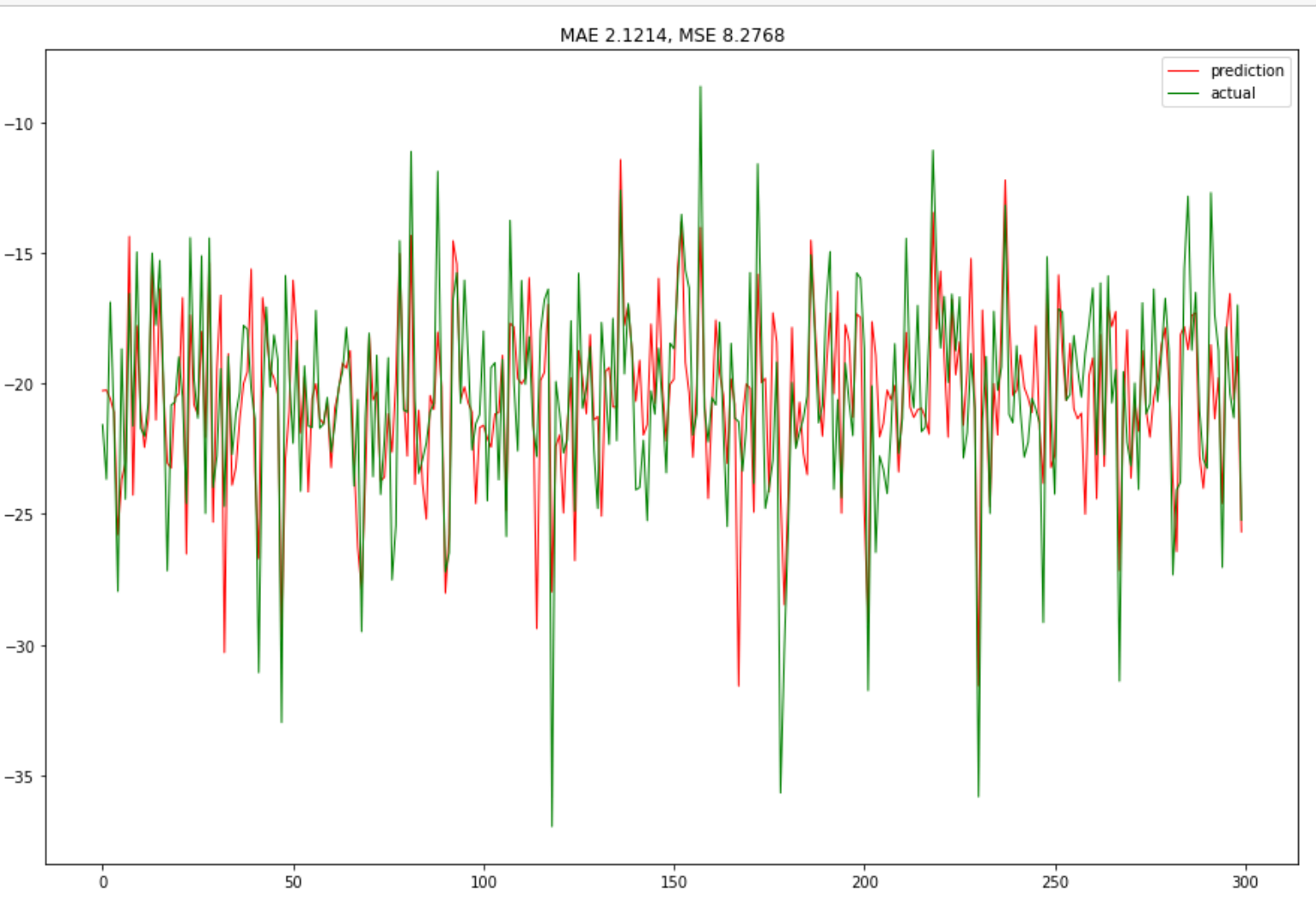
In [196]:
```
ridge = RidgeCV(cv=5)
ridge.fit(X_train, y_train)
evaluation(ridge, X_test, y_test)
```




```
MAE score: 2.3763
MSE score: 10.0647
```

Extra tree regressor ---- MAE AND MSE calculation

In [154]:
```
et = ExtraTreesRegressor(n_estimators=155)

et.fit(X_train, y_train)
evaluation(et, X_test, y_test)
```




```
MAE score: 2.1214
MSE score: 8.2768
```

Word2vec

In [236]:
```
model = word2vec.Word2Vec.load("/gdrive/My Drive/Dataset/AICrowd/Covid/model_300dim.pkl")
```

```
/usr/local/lib/python3.6/dist-packages/smart_open/smart_open_lib.py:253: UserWarning: This fu
nction is deprecated, use smart_open.open instead. See the migration notes for details: http
s://github.com/RaRe-Technologies/smart_open/blob/master/README.rst#migrating-to-the-new-open-
function
  'See the migration notes for details: %s' % _MIGRATION_NOTES_URL
```

In [237]:
```
df2 = pd.read_csv("/gdrive/My Drive/Dataset/AICrowd/Covid/train.csv")
print(df2.head())

target = df2['Binding Affinity']
df2.drop(columns='Binding Affinity',inplace=True)
```

```
           SMILES sequence  Binding Affinity
0     CCNC(C)C(NC)c1ccccc1          -18.0861
1         CONC(=O)c1cncnc1          -17.5783
2    CCNC1CCCN(Cc2ccsc2)C1          -20.3645
3  CC(NC(=O)CSCCN)c1ccccc1          -19.3144
4     CCC(CS)CN(C)c1ccccc1          -15.8451
```

In [238]:
```
df2['mol'] = df2['SMILES sequence'].apply(lambda x: Chem.MolFromSmiles(x))

#Constructing sentences
df2['sentence'] = df2.apply(lambda x: MolSentence(mol2alt_sentence(x['mol'], 1)), axis=1)

#Extracting embeddings to a numpy.array
#Note that we always should mark unseen='UNK' in sentence2vec() so that model is taught how
 to handle unknown substructures
df2['mol2vec'] = [DfVec(x) for x in sentences2vec(df2['sentence'], model, unseen='UNK')]
X = np.array([x.vec for x in df2['mol2vec']])
y = target.values

print(X.shape)
```

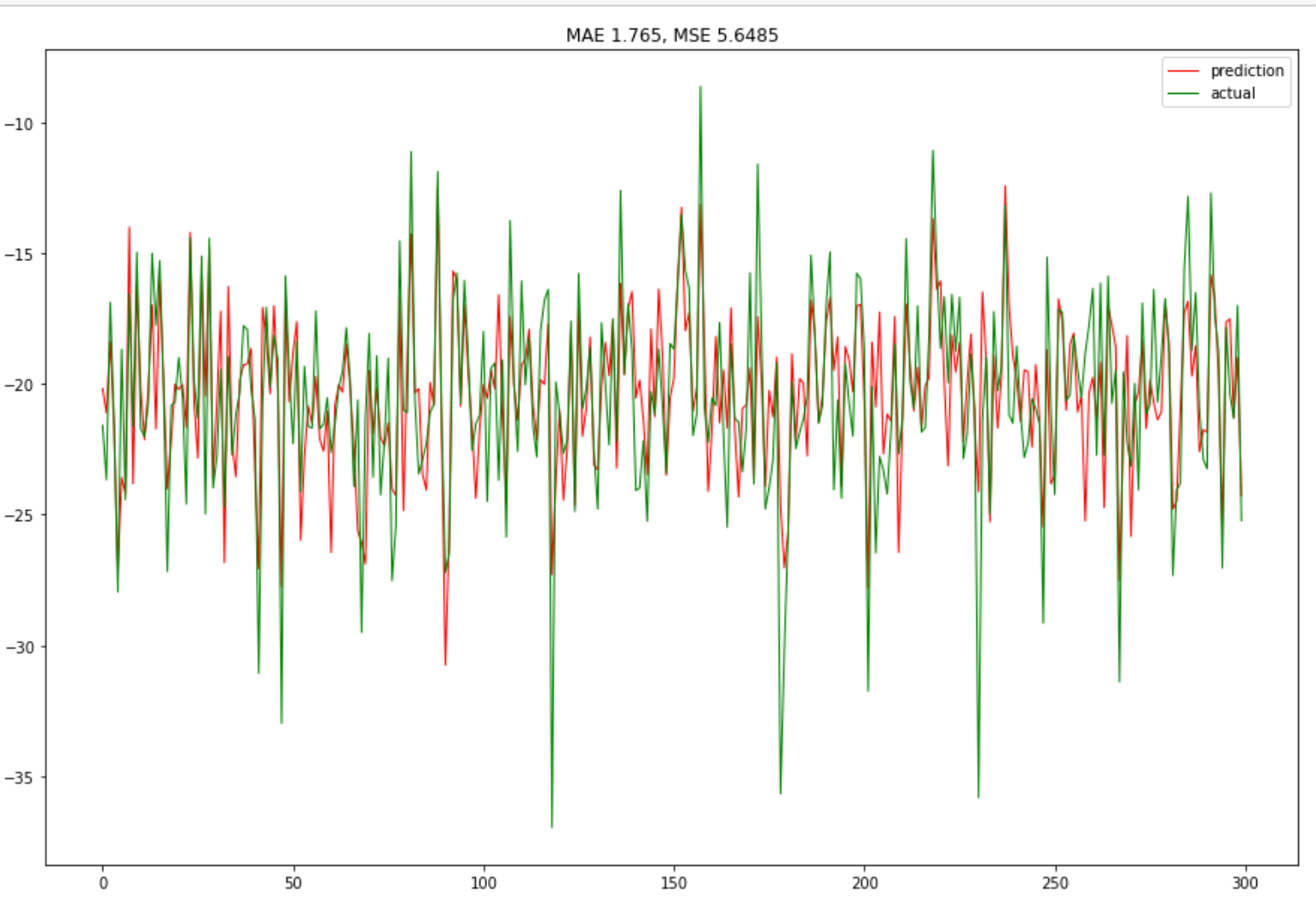
```
(9000, 300)
```

In [247]:
```
mdf = pd.DataFrame(X)
new_df = pd.concat((mdf, train_df), axis=1)
new_df.drop(columns=["num_of_atoms", "num_of_heavy_atoms", "num_of_C_atoms", "num_of_O_atom
s", "num_of_N_atoms", "num_of_Cl_atoms", "tpsa", "mol_w", "num_valence_electrons", "num_hete
roatoms"], inplace=True)
print(new_df.shape)

X_train, X_test, y_train, y_test = train_test_split(new_df, y, test_size=.1, random_state=1)
```

```
(9000, 300)
```

Using ridgecv after mol-to-vec conversion ------MAE AND MSE calculation

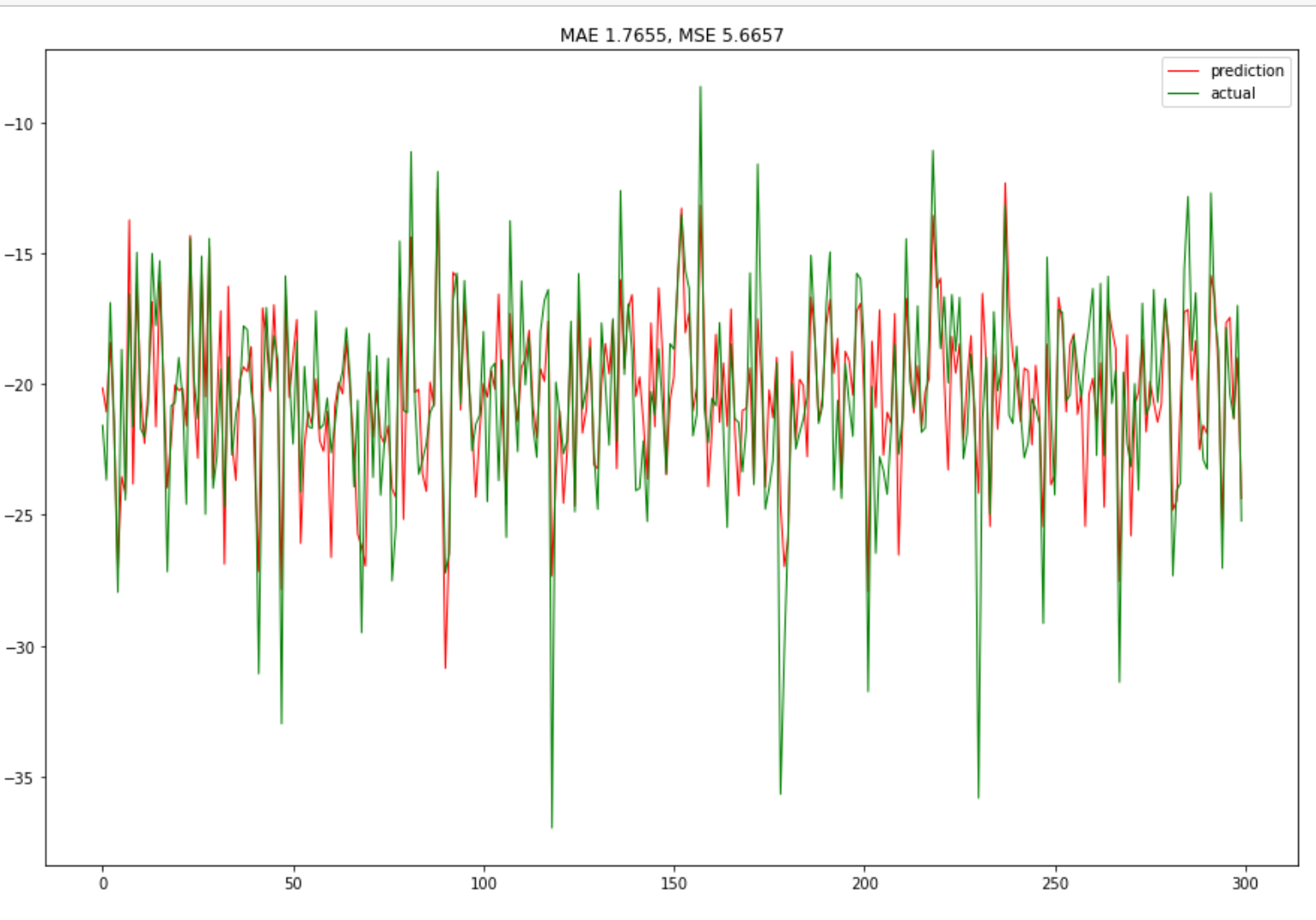
In [248]:
```
ridge = RidgeCV(cv=5)
ridge.fit(X_train, y_train)
evaluation(ridge, X_test, y_test)
```




```
MAE score: 1.765
MSE score: 5.6485
```

Linear regression ----- MAE AND MSE calculation

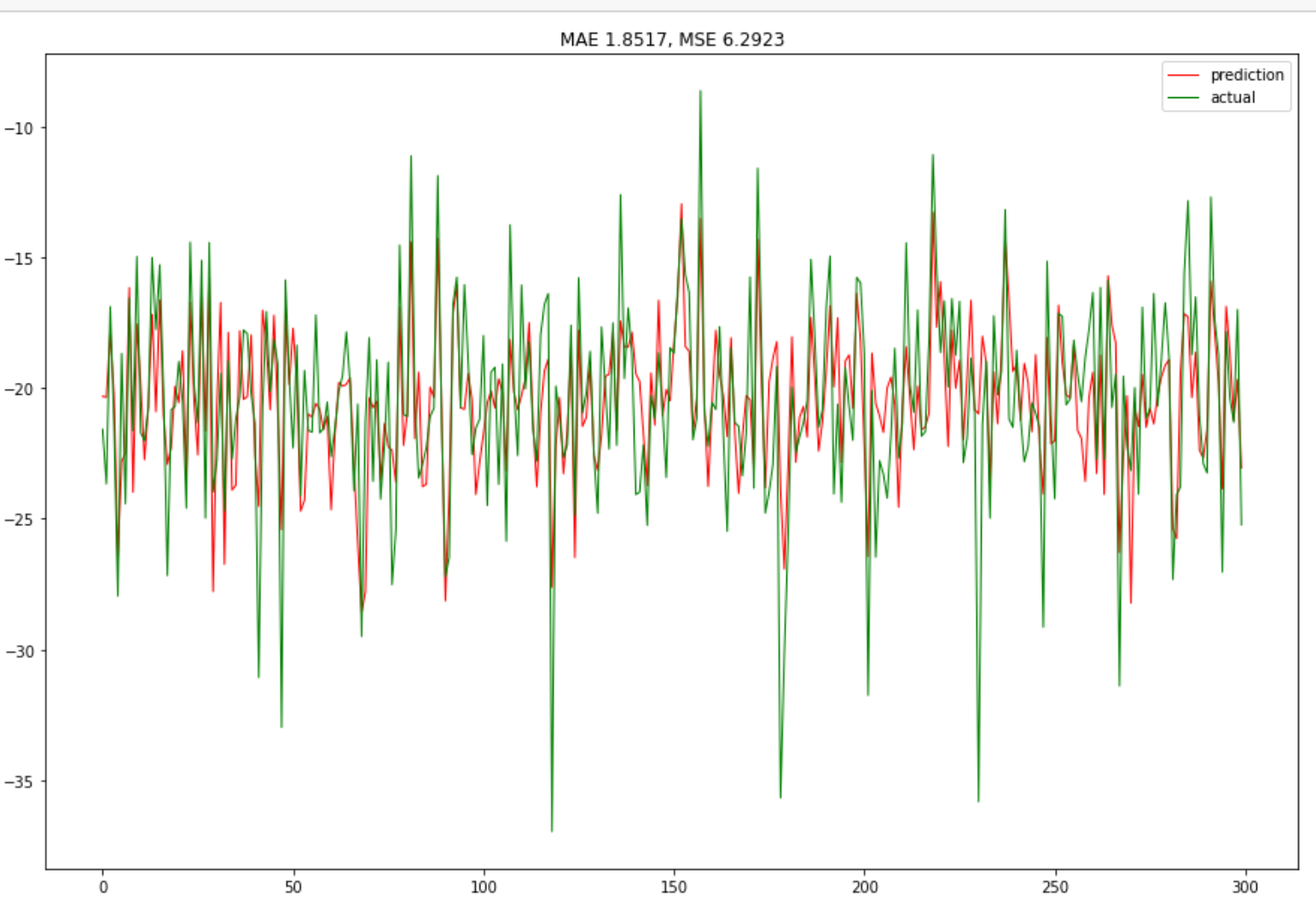
In [161]:
```
regr = LinearRegression()

regr.fit(X_train, y_train)
evaluation(regr, X_test, y_test)
```




```
MAE score: 1.7655
MSE score: 5.6657
```

using extraTreeRegressor --- MAE AND MSE calculation

In [162]:
```
from time import time

start_time = time()

et = ExtraTreesRegressor(n_estimators=150, n_jobs=-1)

et.fit(X_train, y_train)
evaluation(et, X_test, y_test)

print("Time taken : {}".format(time() - start_time))
```




```
MAE score: 1.8517
MSE score: 6.2923
Time taken : 63.956183433532715
```

Testing

In [0]:
```
df1 = pd.read_csv('/gdrive/My Drive/Dataset/AICrowd/Covid/test.csv')
df1.drop(columns="Binding Affinity", inplace=True)
```

In [0]:
```
df1['mol'] = df1['SMILES sequence'].apply(lambda x: Chem.MolFromSmiles(x))
df1['sentence'] = df1.apply(lambda x: MolSentence(mol2alt_sentence(x['mol'], 1)), axis=1)
df1['mol2vec'] = [DfVec(x) for x in sentences2vec(df1['sentence'], model, unseen='UNK')]

X_test = np.array([x.vec for x in df1['mol2vec']])
```

In [251]:
```
X_test.shape
```

Out[251]: (2500, 300)

In [255]:
```
df1["Binding Affinity"] = ridge.predict(X_test)
df1.drop(columns=["mol", "sentence", "mol2vec"], inplace=True)
df1.head()
```

Out[255]:

	SMILES sequence	Binding Affinity
0	Cc1ccc(C2CNCCN2C)cc1	-22.368298
1	CCOC(CO)c1ccccc1	-14.272846
2	CC(=O)Nc1cnn(C)n1	-23.818623
3	CCC(C)NCc1ncccn1	-20.645805
4	CC(C)=C1CC(N)C1	-20.564342

In [0]:
```
df1.to_csv("/gdrive/My Drive/Dataset/AICrowd/Covid/submission.csv", index=False)
```