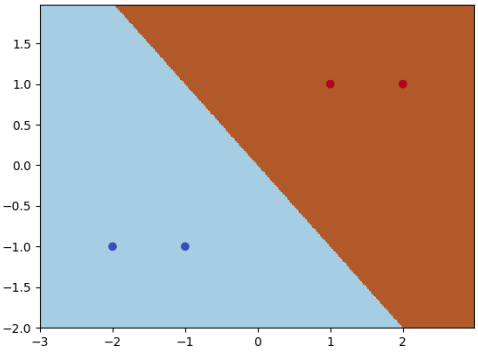
Support Vector Machines, Practical Session

Abdulrasaq, Surajudeen.* and Ezukwoke, Ifeanyi Kenneth.**

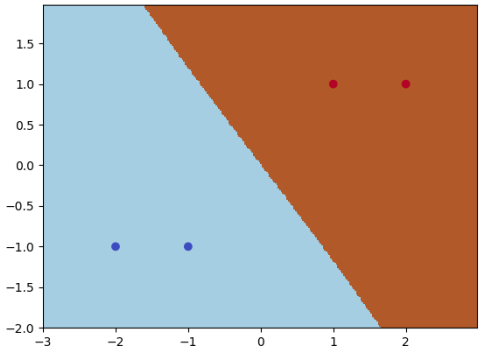
*MLDM M1

Exercise 2: SVM with Scikit-Learn

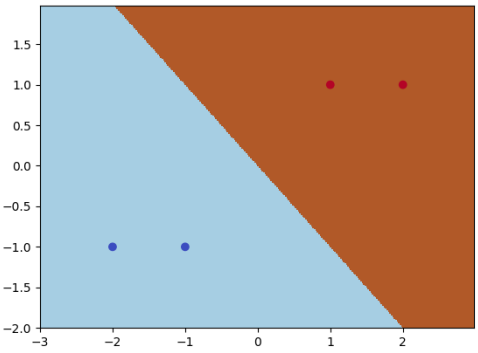
```
 # Numpy is a package for scientific computing.
import numpy as np
#we create 4 examples
X = np.array([[-1, -1], [-2, -1], [1, 1], [2, 1]])
y = np.array([-1, -1, 1, 1])
from sklearn.svm import SVC
#we create a SVM with default parameters
classif=SVC()
#we learn the model according to given data
classif.fit(X,y)
#prediction on a new sample
res=classif.predict([[-0.8, -1]])
print(res);
#We will now create a 2d graphic to illustrate the learning result
import matplotlib.pyplot as plt #the library for plotting
#we create a mesh to plot in
h = .02 # grid step
x_min= X[:, 0].min() - 1
x_max= X[:, 0].max() + 1
y_min = X[:, 1].min() - 1
y_max = X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
np.arange(y_min, y_max, h))
#the grid is created, the intersections are in xx and yy
mysvc= SVC(kernel='rbf', C = 2.0)
mysvc.fit(X,y)
# we predict all the grid
Z2d = mysvc.predict(np.c_[xx.ravel(),yy.ravel()])
Z2d=Z2d.reshape(xx.shape)
plt.figure()
plt.pcolormesh(xx,yy,Z2d, cmap=plt.cm.Paired)
# We plot also the training points
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.coolwarm)
plt.show()
```



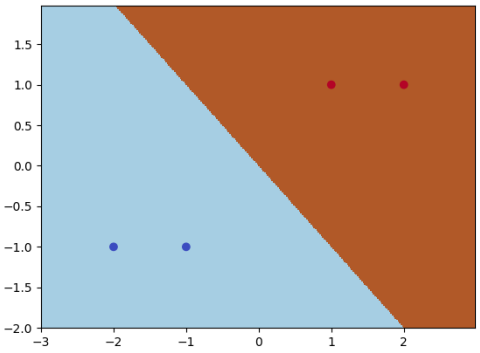
(a) (linear Kernel and C = 2.0)



(b) (RBF Kernel and C = 2.0)



(a) (poly Kernel and C = 1.0))



(b) (sigmoid Kernel and C =1.0)

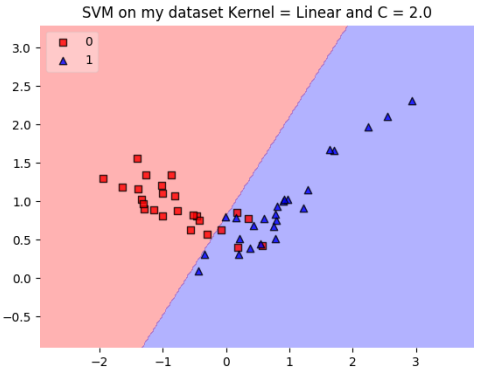
Observation

(C parameter regulate the margin, the higher the C the tighter the model fit, lower C results in flat outputs, while gamma parameter has no relationship with C from my observation)

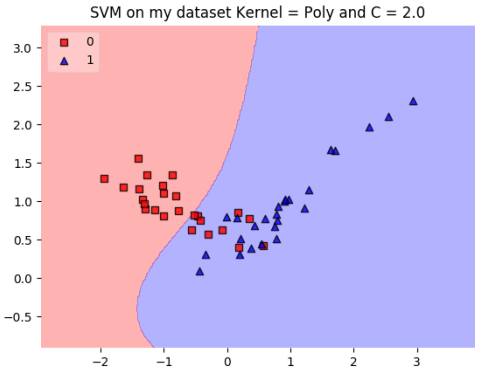
Dataset Generation(Exercise 3.1)

Random datasets

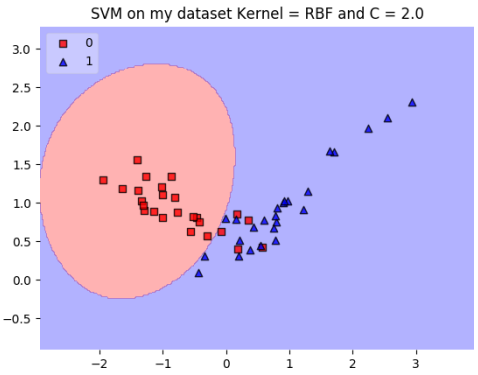
```
from sklearn import svm
from sklearn.cross_validation import train_test_split
from sklearn.model_selection import LeaveOneOut
from sklearn.grid_search import GridSearchCV
from sklearn.datasets import make_classification
from matplotlib import pyplot as plt
from mlxtend.plotting import plot_decision_regions

#ceate our random data
X, Y = make_classification(n_samples=50,n_features=3, n_redundant=0, n_informative=2, random_state=2,
    n_clusters_per_class=1)
#split our data into train and test data
X = X[:, [0,2]]
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=.3, random_state=0)


svc = svm.SVC()

#parameters to tune the data
parameter_candidates = [{'C': [1, 100, 1000], 'kernel': ['linear']},
{'C': [1, 100, 1000], 'kernel': ['sigmoid']},
{'C': [1, 100, 1000], 'kernel': ['poly']},
{'C': [1, 100, 1000],'gamma': [0.001, 0.0001], 'kernel': ['rbf']}
]
#using the leave_one_out cross validation
leave_out = LeaveOneOut()
leave_out.get_n_splits(X)

#split our data into train test sub data
for learn,test in leave_out.split(X):
#split model
X_train, X_test = X[learn], X[test]
Y_train, Y_test = Y[learn], Y[test]
# Create a classifier object with the classifier and parameter candidates
clf = GridSearchCV(estimator=svc, param_grid=parameter_candidates, n_jobs=-1)

# Train the classifier on data1's feature and target data
clf.fit(X_train, Y_train)

# View the accuracy score
print('Best score for data1:', clf.best_score_)
# View the best parameters for the model found using grid search
print('Best C:',clf.best_estimator_.C)
```



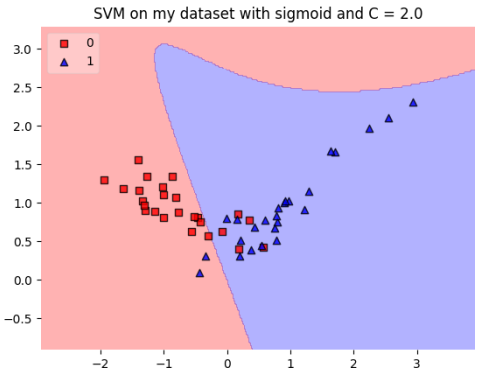
(a) (Score = 70)



(b) (Score = 60)



(a) (Score = 70))



(b) (Score = 70)

Observation

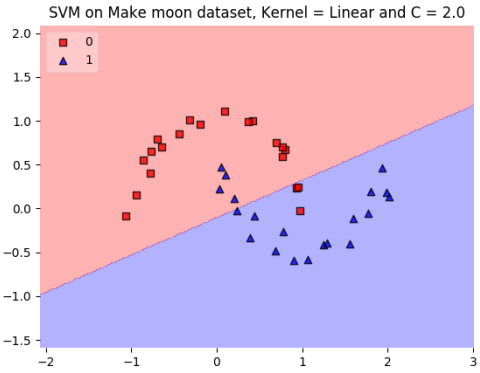
(RBF, SIGMOID AND LINEAR all gave the same score while polynomial seems to give the lowest results, but the model choose Best Kernel: sigmoid, Best Gamma: auto)

```
print('Best Kernel:',clf.best_estimator_.kernel)
print('Best Gamma:',clf.best_estimator_.gamma)


plot_decision_regions(X, Y, clf, res=0.02, legend=2)
plt.title('SVM on my dataset Kernel = Poly and C = 2.0')
plt.show()
```

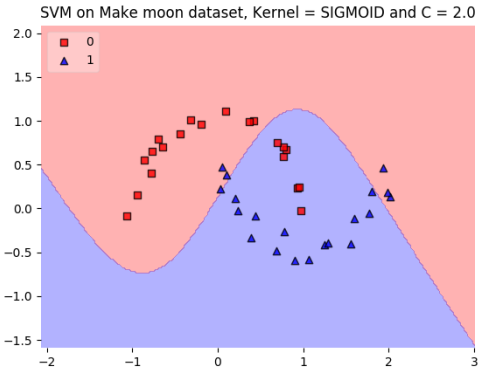
Existing datasets (Exercise 3.2)

```
from sklearn.datasets import make_moons, make_classification
from matplotlib import pyplot as plt
from sklearn import cross_validation, svm
from mlxtend.plotting import plot_decision_regions
from sklearn.model_selection import train_test_split
from sklearn.model_selection import LeaveOneOut
from sklearn.grid_search import GridSearchCV

X, Y = make_moons(noise=0.1, random_state=1, n_samples= 40)

#split our data into train and test data
#X = X[:, [0,2]]
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=.3, random_state=0)


svc = svm.SVC()

#parameters to tune the data
```

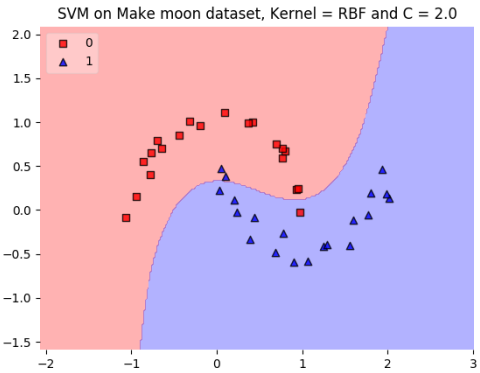
```
parameter_candidates = [{'C': [1, 100, 1000], 'kernel': ['linear']},
{'C': [1, 100, 1000], 'kernel': ['sigmoid']},
{'C': [1, 100, 1000], 'kernel': ['poly']},
{'C': [1, 100, 1000],'gamma': [0.001, 0.0001], 'kernel': ['rbf']}
]
#using the leave_one_out cross validation
leave_out = LeaveOneOut()
leave_out.get_n_splits(X)

#split our data into train test sub data
for learn,test in leave_out.split(X):
#split model
X_train, X_test = X[learn], X[test]
Y_train, Y_test = Y[learn], Y[test]
# Create a classifier object with the classifier and parameter candidates
clf = GridSearchCV(estimator=svc, param_grid=parameter_candidates, n_jobs=-1)

# Train the classifier on data1's feature and target data
clf.fit(X_train, Y_train)

# View the accuracy score
print('Best score for data1:', clf.best_score_)
# View the best parameters for the model found using grid search
print('Best C:',clf.best_estimator_.C)
print('Best Kernel:',clf.best_estimator_.kernel)
print('Best Gamma:',clf.best_estimator_.gamma)


plot_decision_regions(X, Y, clf, res=0.02, legend=2)
plt.title('SVM on my dataset Kernel = Poly and C = 2.0')
plt.show()
```



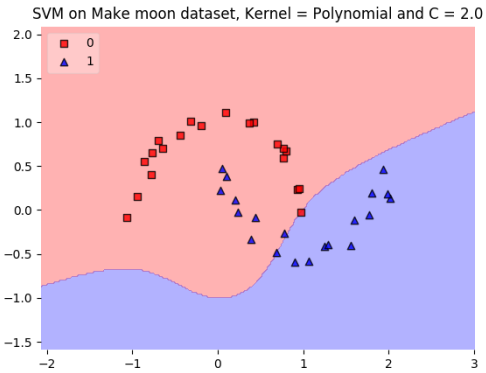
(a) (Score = 86.5)



(b) (Score = 81)



(a) (Score = 78.2))



(b) (Score = 50)

Observation

(LINEAR Score is 86 making it the best as selected by the model (Best Kernel: linear Best Gamma: auto), SIGMOID AND RBF also score high while once again polynomial score the lowest.)

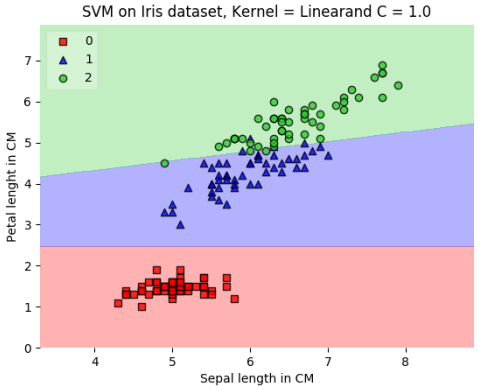
Make Moon Dataset (sklearn)

Iris Dataset (sklearn)

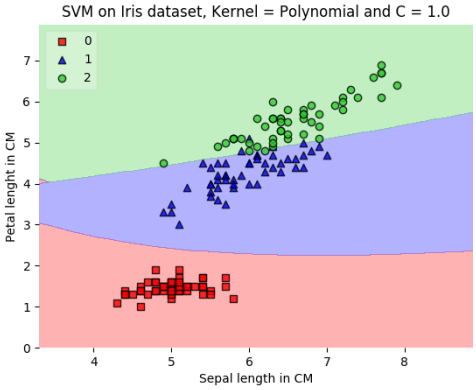
```
from sklearn.datasets import load_iris
from matplotlib import pyplot as plt
from sklearn import cross_validation, svm
from mlxtend.plotting import plot_decision_regions
from sklearn.model_selection import train_test_split
from sklearn.model_selection import LeaveOneOut
from sklearn.grid_search import GridSearchCV

# Lets get to know the data

my_data = load_iris()
print(my_data.data)
print(my_data.target)
print(my_data.target_names)
print(my_data.feature_names)
print(my_data.DESCR)

X = my_data.data[:,[0,2]]
Y = my_data.target


svc = svm.SVC()

#parameters to tune the data
parameter_candidates = [{'C': [1, 100, 1000], 'kernel': ['linear']},
{'C': [1, 100, 1000], 'kernel': ['sigmoid']},
{'C': [1, 100, 1000], 'kernel': ['poly']},
{'C': [1, 100, 1000],'gamma': [0.001, 0.0001], 'kernel': ['rbf']}
]
#using the leave_one_out cross validation
leave_out = LeaveOneOut()
leave_out.get_n_splits(X)

#split our data into train test sub data
for learn,test in leave_out.split(X):
#split model
X_train, X_test = X[learn], X[test]
Y_train, Y_test = Y[learn], Y[test]
# Create a classifier object with the classifier and parameter candidates
clf = GridSearchCV(estimator=svc, param_grid=parameter_candidates, n_jobs=-1)

# Train the classifier on data1's feature and target data
clf.fit(X_train, Y_train)

# View the accuracy score
print('Best score for data1:', clf.best_score_)
# View the best parameters for the model found using grid search
print('Best C:',clf.best_estimator_.C)
print('Best Kernel:',clf.best_estimator_.kernel)
print('Best Gamma:',clf.best_estimator_.gamma)


# Plotting decision regions
plot_decision_regions(X, Y, clf, res=0.02, legend=2)
plt.xlabel('Sepal length in CM')
plt.ylabel('Petal lenght in CM')
plt.title('SVM on Iris dataset, Kernel = Polynomial and C = 1.0')
plt.show()
```

REAL Dataset (Ozone) (Exercise)

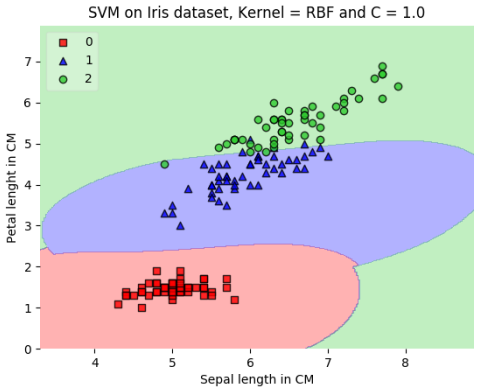
```
import os
os.chdir("/home/suraj/PycharmProjects/practical2_amaury")
import pandas as pd
from sklearn.preprocessing import StandardScaler
```



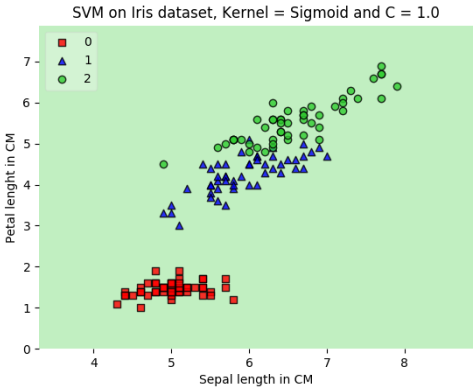
(a) (Score = 96.5)



(b) (Score = 95)



(a) (Score = 94.2))



(b) (Score = 79)

Observation

(LINEAR Score is 96 making it the best as selected by the model (Best Kernel: linear Best Gamma: auto), POLYNOMIAL AND RBF also score high while once again sigmoid score the lowest.)

```
import pandas as pd
from math import sqrt, log
import numpy as np
from sklearn.cross_validation import train_test_split
from sklearn.model_selection import LeaveOneOut
from sklearn.grid_search import GridSearchCV
from sklearn.datasets import make_classification
from sklearn import linear_model

# #import the ozone data file
input_file = "ozone.dat"
df = pd.read_csv(input_file, sep = " ", header = 0)
df.head()
print(df)

df['STATION'] = pd.Categorical(df['STATION'], ordered = False)
df['JOUR'] = pd.Categorical(df['JOUR'], ordered = False)
df['O3obs'] = pd.Categorical(df['O3obs'], ordered = False)
df.dtypes
df.describe()


# #preprocessing our data--> data transformation
# from math import sqrt, log

df["SRMH2O"] = df["RMH2O"].map(lambda x : sqrt(x))
df["LNO2"] = df["NO2"].map(lambda x : log(x))
df["LNO"] = df["NO"].map(lambda x: log(x))
df
#Now we delete the untransformed data
del df['RMH2O']
del df['NO2']
del df['NO']
df
df.hist()

df.head()
df0 = pd.get_dummies(df[['JOUR', 'STATION']])
del df0['JOUR_0']
df0.head()

#Now leta map the data

df['Target'] = df['O3obs'].map(lambda x: x > 150)
df.head()

XZ = df[["MOCAGE", "TEMPE", "VentMOD", "VentANG", "SRMH2O", "LNO2", "LNO"]]
X = pd.concat([df0,XZ], axis = 1)
X

#convert the True False to zeros and ones
Y = df['Target'].map(lambda x: int(x))
#The ozone classification target
Y_ozone = df['O3obs']
Y_ozone.hist()

#ceate our random data
#split our data into train and test data
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=.3, random_state=0)
#X_train, X_test, Y_train, Y_test = train_test_split(X, Y_ozone, test_size=.3, random_state=0)

h= 0.02
svc = linear_model.Lasso()

#parameters to tune the data
parameter_candidates = [{'alpha': [0.05, 0.1,0.2, 0.3, 0.4, 0.5, 1]}]
#using the leave_one_out cross validation
leave_out = LeaveOneOut()
leave_out.get_n_splits(X)
```

```
#split our data into train test sub data
for learn,test in leave_out.split(X):
#split model
X_train = X.loc[learn]
Y_train = Y.loc[learn]
X_test = X.loc[test]
Y_test = Y.loc[test]
Yoz_train = Y_ozone.loc[learn]
Yoz_test = Y_ozone.loc[test]
# Create a classifier object with the classifier and parameter candidates
clf = GridSearchCV(estimator=svc, param_grid=parameter_candidates, n_jobs=-1)
#clf1 = GridSearchCV(estimator=svc, param_grid=parameter_candidates, n_jobs=-1)

# Train the classifier on data1's feature and target data
clf.fit(X_train, Y_train)
#clf1.fit(X_train, Yoz_train)

# View the accuracy score
print("===BEST RESULT FOR THE TARGET LABEL===")
#print('Best score for data1:', clf.best_score_)
# View the best parameters for the model found using grid search
print('Best alpha:',clf.best_params_['alpha'])
print('Best score:',clf.best_score_)
```

Observation

We iterated over the different kernel which took a bit of time. In the end got the following result and the conclusions

> The best score : 89Best C: 5 Best kernel: Linear Best Gamma: auto The best score : 81Best C: 1 Best kernel: sigmoid Best Gamma: auto The best score : 84Best C: 1 Best kernel: rbf Best Gamma: auto The best score : 82Best C: 1 Best kernel: poly Best Gamma: auto

From the above result, Linear kernel gave the best result as seen above followed next by the rbf kernel.

Using Lasso Regression for Prediction

```
import os
os.chdir("/home/suraj/PycharmProjects/practical2_amaury")
import pandas as pd
from math import sqrt, log
import numpy as np
from sklearn.model_selection import KFold
from sklearn.cross_validation import train_test_split
from matplotlib import pyplot as plt
from sklearn import linear_model

# #import the ozone data file
input_file = "ozone.dat"
df = pd.read_csv(input_file, sep = " ", header = 0)
df.head()
print(df)

df['STATION'] = pd.Categorical(df['STATION'], ordered = False)
df['JOUR'] = pd.Categorical(df['JOUR'], ordered = False)
df['O3obs'] = pd.Categorical(df['O3obs'], ordered = False)
df.dtypes
df.describe()


# #preprocessing our data--> data transformation
# from math import sqrt, log

df["SRMH2O"] = df["RMH2O"].map(lambda x : sqrt(x))
df["LNO2"] = df["NO2"].map(lambda x : log(x))
df["LNO"] = df["NO"].map(lambda x: log(x))
df
#Now we delete the untransformed data
del df['RMH2O']
del df['NO2']
```

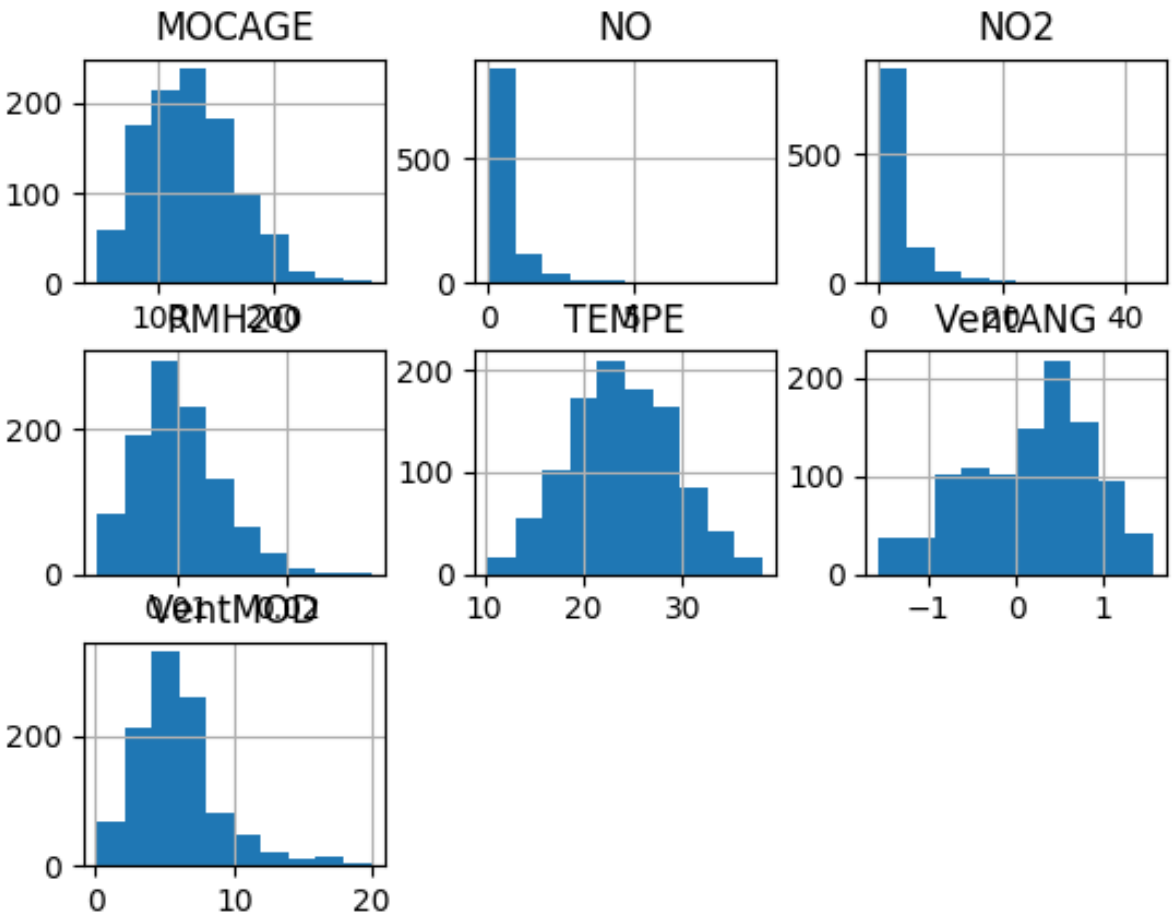


Figure 9: *Shows ozone datasets.*

```
del df['NO']
df
df.hist()

df.head()
df0 = pd.get_dummies(df[['JOUR', 'STATION']])
del df0['JOUR_0']
df0.head()

#Now leta map the data

df['Target'] = df['O3obs'].map(lambda x: x > 150)
df.head()

XZ = df[["MOCAGE", "TEMPE", "VentMOD", "VentANG", "SRMH2O", "LNO2", "LNO"]]
X = np.array(pd.concat([df0,XZ], axis = 1))


#convert the True False to zeros and ones
Y = np.array(df['Target'].map(lambda x: int(x)))
#The ozone classification target
Y_ozone = df['O3obs']
Y_ozone.hist()

#ceate our random data
#split our data into train and test data
lasso = linear_model.Lasso()
X_train, X_test, Y_train, Y_test = train_test_split(X, Y_ozone, test_size=0.2, random_state=0)
#clf = svm.SVC()
lasso.fit(X_train,Y_train)
my_score = lasso.score(X_test,Y_test)
pred = lasso.predict(Y[:,1])
#print(my_score)
print(pred)
plt.plot(pred)

plt.show()
```

Observations

Best alpha: 0.05 Best score: 0.24176487171826797
PEAK PREDICTION OF OZONE: 194.12238851360664

CONCLUSION

Using Lasso regression we predicted that our ozone level decreased by more than 35The meaning of this is that the activities of industries have increased environmental pollution emission of greenhouse gases that have plagued the ozone. Consequently leading to a significant reduction in the ozone layer.