ME4131D

Machine Learning for Data Science and Analytics

Course Project

Prediction of Ozone Days

Submitted by:

John Joji Melel	**B180329ME**
Gayathri S	**B180083ME**
Hareendran V R	**B180073ME**
U Tarun Kumar	**B180774ME**

Department of Mechanical Engineering
NATIONAL INSTITUTE OF TECHNOLOGY CALICUT

CONTENTS

1. Introduction

Ozone levels are a very important factor which affect the weather at particular location at a particular time. If the ozone level is less then it can cause many issues like sunburns. Therefore, it is necessary to predict the ozone levels at a particular location based on factors like temperature, wind speed, humidity, etc. in order to make sure that the weather is safe enough for outdoor activities. In this project, we have predicted whether a day is ozone day or not based on the past weather data available for the location.

2. Problem Definition

The problem considered is to determine whether a day will be ozone day or not based on factors like temperature, wind speed, etc. When the ozone level drops below a certain point, it is considered as ozone day. Otherwise it is considered to be a normal day. Therefore, the output is a binary variable where 0 denotes normal day and 1 denotes ozone day. The features that determine the ozone level are:

- Temperature (T)
- Wind speed (WS)
- Relative humidity (RH)
- U wind in east-west direction (U)
- V wind in north-south direction (V)
- Geopotential height (HT)

These properties are measured at different heights namely 850 hPa (low altitude), 700 hPa (3100 m) and 50 hPa (5500 m). Along with these above-mentioned properties, the following properties also affect the ozone levels.

- K-index - A measure of the thunderstorm potential based on vertical temperature lapse rate, moisture content of the lower atmosphere, and the vertical extent of the moist layer. (KI)
- T-Totals -An index used to assess storm strength. (TT)
- Sea level pressure (SLP)
- Sea level pressure from previous day (SLP_)
- Precipitation (Precp)
- Peak wind speed (WS_PK)
- Average wind speed (WS_AV)
- Peak temperature (T_PK)
- Average temperature (T_AV)

The problem is to predict whether a day is ozone day or not based on the past data available. Therefore, the output variable is a binary variable, that is, if it is an ozone day the value of the output will be on. If it is a normal day, the value will be 0.

3. Literature Review

In the work done by Kun Zhang and Wei Fan, they have helped Texas Commission on Environmental Quantity (TCEQ) to build highly accurate ozone level alarm forecasting models for the Houston area, where these technical difficulties come together in one single problem. The main characteristics of this problem are:

- The dataset is sparse (72 features, and 2% or 5% positives depending on the criteria of "ozone days")
- Evolving over time from year to year
- Limited in collected data size (7 years or around 2500 data entries)
- Contains a large number of irrelevant features
- It is biased in terms of "sample selection bias"
- The true model is stochastic as a function of measurable factors.

The prediction accuracy of the chosen approach (bagging probabilistic decision trees and random decision trees) is 20% higher in recall (correctly detects 1 to 3 more ozone days, depending on the year) and 10% higher in precision (15 to 30 fewer false alarm days per year) than state-of-the-art methods used by air quality control scientists. From their work, it can be inferred that the two straight-forward non-parametric methods (bagging probabilistic decision trees and random decision trees) can provide significantly more accurate and reliable solutions than a number of sophisticated and well-known algorithms, such as SVM and AdaBoost.

4. Description of Dataset

The dataset considered is a multivariate, sequential time-series collected from 1998 to 2004 at the Houston, Galveston and Brazoria counties in the USA. The dataset has 73 attributes and 2536 instances. The attributes are conditions like Temperature (T), Wind speed (WS), Relative humidity (RH), U wind in east-west direction (U), V wind in north-south direction (V) and Geopotential height (HT) at different locations measured at different times. The other features include K-index (KI), T-Totals (TT), Sea level pressure (SLP), Sea level pressure from previous day (SLP_), Precipitation (Precp), Peak wind speed (WS_PK), Average wind speed (WS_AV), Peak temperature (T_PK), Average temperature (T_AV). These features are used to determine whether a day is ozone day or not.

5. Methodology

- **Logistic Regression**
  The Logistic Regression model was made taking all 72 columns and maximum iterations as 10,000. We took an 80:20 ratio of Train:Test data and trained the model for all the 2545 rows. The model showed 0.9407 score for train data and 0.9613 score for test data.

- **Support Vector Machines**
  The Support Vector Machines model was made taking all 72 columns, regularization parameter as 0.0001, and maximum iterations as 100,000. We took an 80:20 ratio of Train:Test data and trained the model for all the 2545 rows. The model showed 0.9329 score for train data and 0.9654 score for test data.

- **Bagging Decision Tree**
  The Bagging Decision Tree model was made taking 50% data for each iteration and with 20 decision trees. We took an 80:20 ratio of Train:Test data and trained the model for all the 2545 rows. The model showed 0.9760 score for train data and 0.9613 score for test data.

- **Random Forest**
  The Bagging Decision Tree model was made using 40 decision trees with a maximum tree depth of 9 and maximum features of 40 out of 72 for classifying each row into a normal day or an ozone day. This was to reduce the level of overfitting in the model. We then took an 80:20 ratio of Train:Test data and trained the model for all the 2545 rows. The model showed 0.9892 score for train data and 0.9634 score for test data.

- **ARIMA**
  After creating the classification model, we used an ARIMA time series prediction model to predict all 72 factor columns for one more week. We analyzed each and every column to identify the optimum combination of p, d, q values and used those values to predict 72 factors for one more week. The optimum (p, q, d) values got from the function is (4,1,1).

Then we used the previously made Logistic Regression, Support Vector Machines, bagging decision tree, and random forest classification models on these newly predicted rows and got the correct predictions.

6. Justification of Algorithms

The five algorithms used are:

- Logistic Regression
- Support Vector Machines
- Bagging Decision Tree
- Random Forest
- ARIMA

- **Logistic Regression**

It is the most basic classification regression algorithm, and is majorly used for binary classification of data based on multiple variables. Logistic regression is a process of modeling the probability of a discrete outcome given an input variable. The most common logistic regression models a binary outcome; something that can take two values such as true/false, yes/no, and so on. The curve has a S shape with the equation

$$p(x) = \frac{1}{1 + e^{-(x-\mu)/s}}$$

- **Support Vector Machines**

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.
The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence the algorithm is termed as Support Vector Machine.

- **Bagging Decision Tree**

Whereas a decision tree might overfit, applying an ensemble learning algorithm like bagging Decision Tree might improve the quality of the prediction model. In bagging, the training data is increased by taking bootstraps from the training data. This means multiple samples are taken (with replacement) from the training data and the model is trained on these sub-datasets. The final prediction is the average over all predictions from each bootstrap sample. It is a better Classification algorithm compared to a traditional Decision Trees.

- **Random Forest**

The Random Forest Algorithm is another frequently used ensemble learning classifier which uses multiple decision trees. The Random Forest classifier is basically a modified bagging algorithm of a Decision Tree that selects the subsets differently. Random Forest works a bit better than Bagging Decision Tree.

- **ARIMA**

ARIMA is an acronym that stands for **AutoRegressive Integrated Moving Average**. It is a class of models that captures a suite of different standard temporal structures in time series data. This algorithm works wonderfully well to predict time series data with good accuracy.

7. Analysis and Results

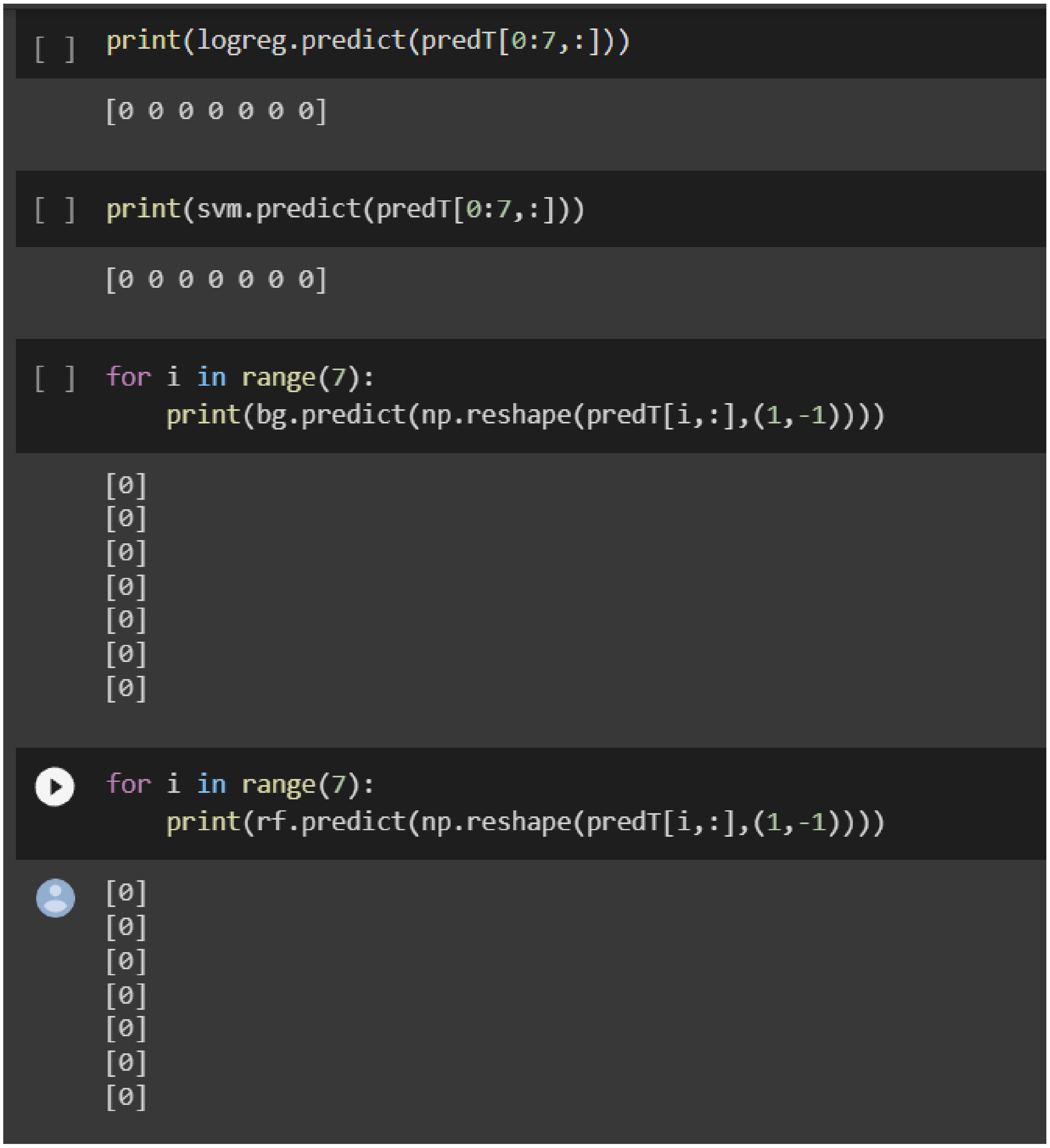
- Logistic Regression
  Score on train: 0.9407443682664055
  Score on test: 0.9613821138211383

- Support Vector Machines
  Score on train: 0.9329089128305583
  Score on test: 0.9654471544715447

- Bagging Decision Tree
  Score on train: 0.9760039177277179
  Score on test: 0.9613821138211383

- Random Forest
  Score on train: 0.9892262487757101
  Score on test: 0.9634146341463414

The models were further used on the future timestamps predicted by ARIMA and gave the same results. There were no ozone days found on the 7 days subsequent to the dataset ending dates.
The output is as shown in figure 1.



```
print(logreg.predict(predT[0:7,:]))
```

```
[0 0 0 0 0 0 0]
```

```
print(svm.predict(predT[0:7,:]))
```

```
[0 0 0 0 0 0 0]
```

```
for i in range(7):
    print(bg.predict(np.reshape(predT[i,:],(1,-1))))
```

```
[0]
[0]
[0]
[0]
[0]
[0]
[0]
```

```
for i in range(7):
    print(rf.predict(np.reshape(predT[i,:],(1,-1))))
```

```
[0]
[0]
[0]
[0]
[0]
[0]
[0]
```

Figure 1. Prediction of ozone days for 7 subsequent dates

8. Discussions and Conclusions

In conclusion, the dataset was predicted in the future using ARIMA algorithm, and four classification algorithms, Logistic Regression, Support Vector Machines, Bagging Decision Tree, and Random Forest were trained on the dataset to finally predict if the predicted days were ozone days or not. The algorithms performed well and gave correct results.
The Support Vector Machines model performed the best out of the four in the test data and Random Forest model performed the best out of the four in the train data.

9. References

- Forecasting Skewed Biased Stochastic Ozone Days: analyses, solutions and beyond by Kun Zhang and Wei Fan. (April 2007)
- https://www.openml.org/search?type=data&sort=runs&id=1487&status=active

10. Annexure

Python Code

```
pip install pmdarima
import numpy as np
import pandas as pd
import io
from pmdarima import auto_arima
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from statsmodels.tsa.arima.model import ARIMA
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings("ignore")

from google.colab import files
ozone = files.upload()

df = pd.read_csv(io.BytesIO(ozone['ozone.csv']))
df.columns
df.info()

x_train = df.iloc[:2042,:72]
y_train = df.iloc[:2042,72]
x_test = df.iloc[2042:,:72]
y_test = df.iloc[2042:,72]

logreg = LogisticRegression(solver='lbfgs', max_iter=10000)
logreg.fit(x_train,y_train)
print("score on train: "+ str(logreg.score(x_train, y_train)))
print("score on test: " + str(logreg.score(x_test, y_test)))

svm = LinearSVC(C=0.0001, max_iter=100000)
svm.fit(x_train, y_train)
print("score on train: "+ str(svm.score(x_train, y_train)))
print("score on test: " + str(svm.score(x_test, y_test)))
```

```
bg =
BaggingClassifier(DecisionTreeClassifier(),max_samples=0.5,max_features=1.0,n_estimators=2
0)
bg.fit(x_train, y_train)
print("score on train: "+ str(bg.score(x_train, y_train)))
print("score on test: " + str(bg.score(x_test, y_test)))

rf = RandomForestClassifier(n_estimators=40, max_features=40, max_depth=9)
rf.fit(x_train, y_train)
print("score on train: "+ str(rf.score(x_train, y_train)))
print("score on test: " + str(rf.score(x_test, y_test)))

for i in range(73):
    stepwise_fit = auto_arima(np.squeeze(df.iloc[i,:].values), trace =True,
suppress_warnings=True)
    stepwise_fit.summary()

print(df.shape)
train=df.iloc[:-7,:]
test=df.iloc[-7:,:]
print(train.shape,test.shape)

start = len(train)
end = len(train) + len(test) - 1
pred = np.zeros((72,7))
for i in range(72):
    model = ARIMA(np.squeeze(df.iloc[i,:].values),order=(4,1,1))
    model = model.fit()
    pred[i,:] = model.predict(start=start,end=end,typ='levels')
predT = pred.T

print(logreg.predict(predT[0:7,:]))
print(svm.predict(predT[0:7,:]))

for i in range(7):
    print(bg.predict(np.reshape(predT[i,:],(1,-1))))

for i in range(7):
    print(rf.predict(np.reshape(predT[i,:],(1,-1))))
```