REPORT ON
MACHINE LEARNING MODEL FOR IOT BASED DOMESTIC VERTICAL FARMING

Developed by Bhashini Alwis (Reg 256) for the partial fulfillment of the capstone project of the Machine learning foundation by Data Science Academy

2022.05.09

First Prediction Model

In this project, I have to make two Machine learning models. One for predicting temperature, PH level and predicting the future values for it. and then, by using them, predict future crop predictions. For predicting temperature, this model can also be used for predicting every other parameter. This model uses Linear regression. Linear Regression is a supervised machine learning technique that predicts continuous output with a constant slope. It is used to forecast values within a continuous range rather than trying to categorize them. There are two major kinds:

Simple Regression

Simple linear regression employs the classic slope-intercept form, with m and b being the variables that our computer will attempt to "learn" in order to provide the most accurate predictions. Our input data is represented by x, and our forecast is represented by y.

$$y=mx+b$$

Multivariable Regression

A more complicated, multi-variable linear equation may look something like this, where w represents the coefficients, or weights, that our model will attempt to learn.

$$f(x,y,z)=w1x+w2y+w3z$$

A more complicated, multi-variable linear equation may look something like this, where w represents the coefficients, or weights, that our model will attempt to learn.

$$Sales=w1Radio+w2TV+w3News$$

This prediction model only uses Simple regression. To make the temperature prediction model, first we need a mock dataset to train. The following figure shows a few data that will be used in the machine learning model.

	A	B	C	D	E	F
1	Date	Temperature	Light Intensity	Co2 Level	Predict Sucess	
2	1/23/2021	2	9	4	63	
3	10/9/2020	6	10	4	31	
4	3/10/2021	1	10	8	84	
5	11/22/2020	4	9	7	92	
6	7/11/2020	5	3	2	90	
7	4/10/2021	2	4	4	93	
8	9/10/2020	2	7	8	68	
9	4/10/2021	3	6	2	56	
10	8/28/2020	3	6	5	74	
11	9/18/2020	4	1	3	24	
12	1/13/2021	1	1	1	77	
13	6/2/2021	10	2	8	90	
14	3/28/2021	4	3	5	91	
15	4/17/2021	7	2	3	85	
16	9/30/2020	7	4	1	74	

Figure 1: First prediction model dataset

Here, temperature, light intensity, and CO_2 level are the parameters that get from Sensors and predict success is a Colom with random data that variant in the range from 1 to 100. And for training purposes, there are 1000 data.1000 data are not enough for a good ML program but for this Demonstration 1000 data give an average rate of accurate predictions.

```
import pandas as pd
import numpy as np
import sklearn
from numpy import ndarray
from sklearn import linear_model
from sklearn.utils import shuffle
import csv
```

Figure 2: Imported libraries

The libraries should be imported. Pandas is mostly used to analyze data. It supports importing data from a variety of file formats, including comma-separated-values, JSON, SQL, and Microsoft Excel. Pandas support a variety of data manipulation operations such as merging, reshaping and selecting, as well as data cleaning and wrangling.

NumPy is a Python package that is used to work with arrays. It also has functions for working with linear algebra, the Fourier transform, and matrices. Also Python's most helpful machine learning library. Sklearn library includes several effective tools for machine learning and statistical modeling, including as classification, regression, clustering, and dimensionality reduction. Later form sklearn liner model is imported. These libraries have to be installed first before importing to the worksheet.

```
data = pd.read_csv("MOCK_DATA.csv", sep=",")

data = data[["Temperature","Predict Sucess"]]
predict = "Predict Sucess"
X = np.array(data.drop([predict], 1))
Y = np.array(data[predict])
```

Figure 3: Making arrays

This is the first part of the code in first line it read the mock dataset that provided by "sep" it means how the data separate from each other, according to figure 1 there are few columns but in here only temperature should by predicted. And we only need past temperature values in second line it filter out the temperature and predict success column, predict success column have just random numbers in range of 1-100.in third line it defined predict success it the column that going to name as predict and that the column that we have to predict. The in 4th and 5th line it have two Arrays that we going to train in X array it have "data.drop" it means it drop the data in predict success column and X is for data sheet Attributes and Y is for the labels Y array have only predict because we only need that column.

```
x_train, x_test, y_train, y_test = sklearn.model_selection.train_test_split(X, Y, test_size=0.1)

linear = linear_model.LinearRegression()

linear.fit(x_train, y_train)
acc = linear.score(x_test, y_test)
print(acc)

prediction=linear.predict(x_test)

for x in range(len(prediction)):
    print(prediction[x])

df = pd.DataFrame(prediction,columns=['Predicts Temperature'])
df.to_csv('test.csv')
```

Figure 4: Train the data and predict

In the second part the X and Y split in to 4 selection s and x_train and y_train have a selection of data in X and Y,x_test and Y_test is used to test our data. Test size means 10% Off our dataset used to test and train, it can be changed according to the accuracy.

Then define the linear model and put the x and y trained into the linear model to train. Define the linear regression model by x_test data, and then by using for loop it and print function it shows the results and in the last lines using panda library it gives an export of prediction values. This is for the future temperature value prediction.

A	B
	Predicts Temperature
0	25.05862415
1	25.53463471
2	25.34127848
3	24.95899404
4	25.15973027
5	25.41950657
6	24.99884608

Crop Prediction Model

For this model it can separate in to 3 main parts to explain easterly

1. Data Pre-Processing
2. Create a Model Training
3. Make Future Predictions

Also, for this model dummy data set has been created as the existing datasets are not proper to use.

	A	B	C	D
1	Date	Temp	Light	Co2
2	2020-06-11	74.83	53.08	2.63
3	2020-06-11	67.48	14.5	2.1
4	2020-06-11	21.33	36.09	40.77
5	2020-06-11	47.95	73.8	58.23
6	2020-06-11	9.56	6.27	75.22
7	2020-06-11	78.75	67	27.59
8	2020-06-12	90.84	15.84	20.05
9	2020-06-12	55.04	30.67	15.76
10	2020-06-12	78.84	77.4	53.6
11	2020-06-13	4.9	1.67	57.9
12	2020-06-13	47.61	84.96	28.68
13	2020-06-14	19.8	21.08	45.41
14	2020-06-14	62.96	14.87	94.27
15	2020-06-14	53.75	67.39	77.42
16	2020-06-15	43.1	49.07	35.98
17	2020-06-15	52.88	56.53	64.46

Figure 5: Dummy dataset

Data set have Date, Temperature light intensity and CO_2 level. These are just dummy data but assume that these values are predicted by last Machine learning Model. Fort his model LSTM (Long Short-Term Memory) Multivariant Time Series to predict the Crop success. The LSTM rectifies a huge issue that recurrent neural networks suffer from: short-memory. Using a series of 'gates,' each with its own RNN, the LSTM manages to keep, forget or ignore data points based on a probabilistic model. LSTMs can also aid in the resolution of bursting and disappearing gradient issues. These issues arise as a result of repetitive weight modifications while a neural network trains. Gradients get greater or smaller when epochs are repeated, and with each modification, it becomes simpler for the network's gradients to compound in either direction. This compounding causes the gradients to be either far too high or far too tiny. While exploding and disappearing gradients are significant drawbacks of standard RNNs, the LSTM design significantly mitigates these difficulties

Data Pre-processing

As the First step libraries should be imported.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import datetime as dt
from datetime import datetime
```

Figure 6: Import libraries

Numpy: Numerical calculations

Matplot: Data resonation data time for time manipulation

Pandas: Data structurers

Keras: Control training

```
dataset_train = pd.read_csv('Data.csv')
cols = list(dataset_train)[1:4]


datelist_train = list(dataset_train['Date'])
datelist_train = [dt.datetime.strptime(date, '%Y-%m-%d').date() for date in datelist_train]

print('Training set shape == {}'.format(dataset_train.shape))
print('All timestamps == {}'.format(len(datelist_train)))
print('Featured selected: {}'.format(cols))
```

Figure 7: Reading dataset

The next step is to preprocess the data by removing all commas and converting data to matrix shape format.

```
dataset_train = dataset_train[cols].astype(str)
for i in cols:
    for j in range(0, len(dataset_train)):
        dataset_train[i][j] = dataset_train[i][j].replace(',', '')

dataset_train = dataset_train.astype(float)

training_set = dataset_train.values

print('Shape of training set == {}.'.format(training_set.shape))
```

Figure 8: Replacing commas

In above figure shows, the conversion of all the lines into string type and then by using for loop commas are replace by the space to separate data. This is done to every selected column in the dataset hence the reason for the loop. The conversion of data to float and the reshaping is done.

```
Shape of training set == (1000, 3).
```

Figure 9: Shape of training set

The above figure tells that out matrix has 1000 rows and three columns.

In this model, temperature, CO_2 level and light intensity might have different scales and it will be not an issue for the first model that created above but, in this model, the plot has to be graph and it's better to make a scaler to precedence from here.

```
sc = StandardScaler()
training_set_scaled = sc.fit_transform(training_set)

sc_predict = StandardScaler()
sc_predict.fit_transform(training_set[:, 0:1])
```

Figure 10: Setting up a scalar

This is the code that has been used to scale all values in the data set a scalar and then this is the end of the step two. Then it's time to make an x_train and y_train like the previous model.

```
X_train = []
y_train = []

n_future = 60
n_past = 90

for i in range(n_past, len(training_set_scaled) - n_future +1):
    X_train.append(training_set_scaled[i - n_past:i, 0:dataset_train.shape[1] - 1])
    y_train.append(training_set_scaled[i + n_future - 1:i + n_future, 0])

X_train, y_train = np.array(X_train), np.array(y_train)

print('X_train shape == {}.'.format(X_train.shape))
print('y_train shape == {}.'.format(y_train.shape))
```

Figure 11: Data training setup

This is the code work for making the data to train. First, as the last model creates X and y train and then define n_future and n_past it means train 90 past days results or values and predict future 60 days and then set the data to trains and in the end convert the x and y train into two NumPy arrays. And print the format of the those to arrays

```
X_train shape == (851, 90, 2).
y_train shape == (851, 1).
```

Figure 12: Data train shape

This is the output of Print functions X train have 851 records and 90 days and 2 1 features and those are predictors. And y train has 851 records and one feature that's the one we going to predict.

Creating a Training Model

Here as the first step, the library is imported, Sequential,dense,LSTM,Dropout from keras and as the optimizer Adam was imported from the keras.

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
from keras.optimizers import Adam
```

Figure 13: Import libraries

```
model = Sequential()
model.add(LSTM(units=64, return_sequences=True, input_shape=(n_past, dataset_train.shape[1]-1)))
model.add(LSTM(units=10, return_sequences=False))
model.add(Dropout(0.25))
model.add(Dense(units=1, activation='linear'))
model.compile(optimizer = Adam(learning_rate=0.01), loss='mean_squared_error')
```

Figure 14: Defining ML model

The training model was defined. First, initialize the neural network based on LSTM.that's the first code line. And then add the first LSTM layer. That having 64 units and put the input shape as the data set shape then the 2nd LSTM layer added with 10 units after that dropout layer Dropout is a regularization method where input and recurrent connections to LSTM units are probabilistically excluded from activation and weight updates while training a network. Then add Dense at last make Optimizer as the Adam and learning rate into 0.01.

```
es = EarlyStopping(monitor='val_loss', min_delta=1e-10, patience=10, verbose=1)
rlr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=10, verbose=1)
mcp = ModelCheckpoint(filepath='weights.h5', monitor='val_loss', verbose=1, save_best_only=True, save_weights_only=True)

tb = TensorBoard('logs')

history = model.fit(X_train, y_train, shuffle=True, epochs=30, callbacks=[es, rlr, mcp, tb], validation_split=0.2, verbose=1, batch_size=256)
```

Figure 15: Setting up training model

As the Final Step in this part, we have to train the x and y, this usually take few seconds to train the data. Meaning of keywords in above figure are:

- **EarlyStopping** - Stop training when a monitored metric has stopped improving.
- **monitor** - quantity to be monitored.
- **min_delta -** minimum change in the monitored quantity to qualify as an improvement, i.e. an absolute change of less than min_delta, will count as no improvement.
- **patience** - number of epochs with no improvement after which training will be stopped.
- **ReduceLROnPlateau** - Reduce learning rate when a metric has stopped improving.
- **factor** - factor by which the learning rate will be reduced. new_lr = lr * factor.

An epoch is a machine learning word that denotes the number of passes the machine learning algorithm has made across the whole training dataset. If the batch size is the whole training dataset, the number of epochs equals the number of iterations.

Make future predictions & visualization

```
datelist_future = pd.date_range(datelist_train[-1], periods=n_future, freq='1d').tolist()

datelist_future_ = []
for this_timestamp in datelist_future:
    datelist_future_.append(this_timestamp.date())

predictions_future = model.predict(X_train[-n_future:])

predictions_train = model.predict(X_train[n_past:])

def datetime_to_timestamp(x):

    return datetime.strptime(x.strftime('%Y%m%d'), '%Y%m%d')
```

Figure 16:Predicting code work

In above figure shows generated days for the predictions by using the dataset data column and then converting Pandas timestamp to datetime object, it's the code in the for a loop. After that, it's time to perform the predictions.

```
y_pred_future = sc_predict.inverse_transform(predictions_future)
y_pred_train = sc_predict.inverse_transform(predictions_train)

PREDICTIONS_FUTURE = pd.DataFrame(y_pred_future, columns=['Open']).set_index(pd.Series(datelist_future))
PREDICTION_TRAIN = pd.DataFrame(y_pred_train, columns=['Open']).set_index(pd.Series(datelist_train[2 * n_past + n_future -1:]))


PREDICTION_TRAIN.index = PREDICTION_TRAIN.index.to_series().apply(datetime_to_timestamp)
```

Figure 17: Make a table using Pandas

Here we use that timestamp and predict that value and by using Pandas data frame made a table for the data. After completing this it's time for the final step, visualizing.

```
dataset_train = pd.DataFrame(dataset_train, columns=cols)
dataset_train.index = datelist_train
dataset_train.index = pd.to_datetime(dataset_train.index)

from pylab import rcParams
rcParams['figure.figsize'] = 14, 5

START_DATE_FOR_PLOTTING = '2012-06-01'

plt.plot(PREDICTIONS_FUTURE.index, PREDICTIONS_FUTURE['Open'], color='r', label='Predicted Growth')
plt.plot(PREDICTION_TRAIN.loc[START_DATE_FOR_PLOTTING:].index, PREDICTION_TRAIN.loc[START_DATE_FOR_PLOTTING:]['Open'], color='orange', label='Training predictions')
##plt.plot(dataset_train.loc[START_DATE_FOR_PLOTTING:].index, dataset_train.loc[START_DATE_FOR_PLOTTING:]['Open'], color='b', label='Actual Stock Price')

plt.axvline(x = min(PREDICTIONS_FUTURE.index), color='green', linewidth=2, linestyle='--')

plt.grid(which='major', color='#cccccc', alpha=0.5)

plt.legend(shadow=True)
plt.title('Growth Prediction', family='Arial', fontsize=12)
plt.xlabel('Timeline', family='Arial', fontsize=10)
plt.ylabel('Stock Price Value', family='Arial', fontsize=10)
plt.xticks(rotation=45, fontsize=8)
plt.show()
```

Figure 18: Visualization

Result

The result of the crop prediction Machine Learning model is Giving Prediction Graph According to the data set.

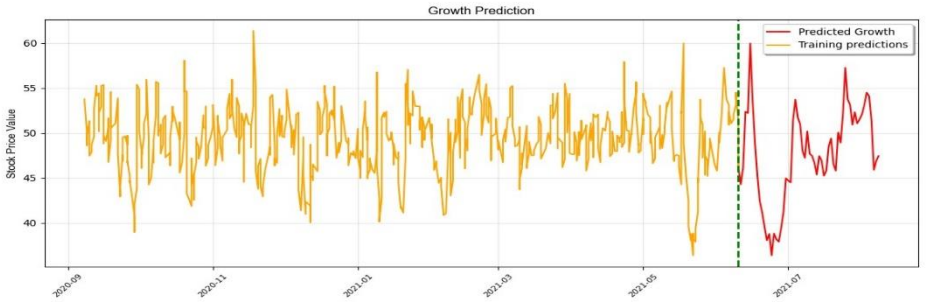

Figure 19: Final output

Yellow shows the previous success and red shows the Future prediction success.