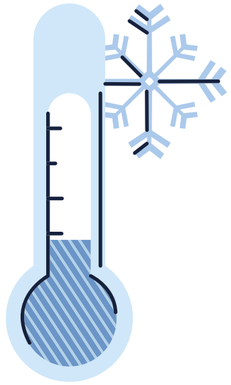
Weather prediction



By: Brute Voltage
Amitasha || Udit || Sai teja

Problem statement

More than 56% of India's population is involved in agriculture. Out of this, more than 20% of the farmers live below the poverty line. A notable reason for this is the presence of middlemen. Fearing deterioration of crops, farmers often get exploited into selling their produce at low prices to middlemen, who then sell at much higher costs. We live in a world where technology is at the heart of our everyday lives. Similar to the transformations in other sectors, technology is sure to shape farming practices. Technology can transform Indian agriculture by addressing challenges related to quality, quantity, distribution and storage.

How it helps to solve the problem?

While various agri-tech solutions are available currently to increase the yield, many barriers have prevented farmers in adopting digital transformation. For example, too much manual work to make the solution function or rely on remote internet accessibility are some of the major challenges. However, tremendous amounts of agricultural data are generated but never used. AI, has come up with a much easier solution to overcome these obstacles. It involves , weather information including weather forecast and soil moisture information, helping farmers to make decisions on water and crop management.

“It is important for a farmer to know when it will rain. If AI can predict the rain accurately and how much and when that will happen, farmers can make meaningful decisions” an expert quoted. It will provide these type of accurate data so that farmers can decide when to irrigate, add fertilisers or delay some decisions. Weather data will also predict diseases.

How it helps to solve the problem?

Here we apply AI, machine learning, and advanced analytics to the farm data to extract valuable insights and automatically generate guidance for smarter decisions. It helps to visualise data and alerts related to critical elements such as weather forecasts, soil conditions, evapotranspiration rates, and crop stress. Farmers could save time, money and also reduce the impact of pests if they understood how and when to spray pesticides. The AI data also provides insights around irrigation, product application, and planting and harvest timing.

Impact parameters

An agricultural weather forecast should refer to all weather elements that immediately affect farm planning or operations. The elements will vary from place to place and from season to season. Normally a weather forecast includes the following parameters.

(a) Amount and type of cloud cover.

(b) Rainfall and snow.

(c) Maximum, minimum and dewpoint temperatures.

(d) Relative humidity.

(e) Wind speed and direction.

(f) Extreme events, such as heatwaves and cold waves, fog, frost, hail, thunderstorms, wind squalls and gales, low-pressure areas, different intensities of depressions, cyclones, and tornadoes.

An agricultural weather forecast should also contain the following information:

(a) Bright hours of sunshine; (b) Solar radiation; (c) Dew; (d) Leaf wetness; (e) Pan evaporation.

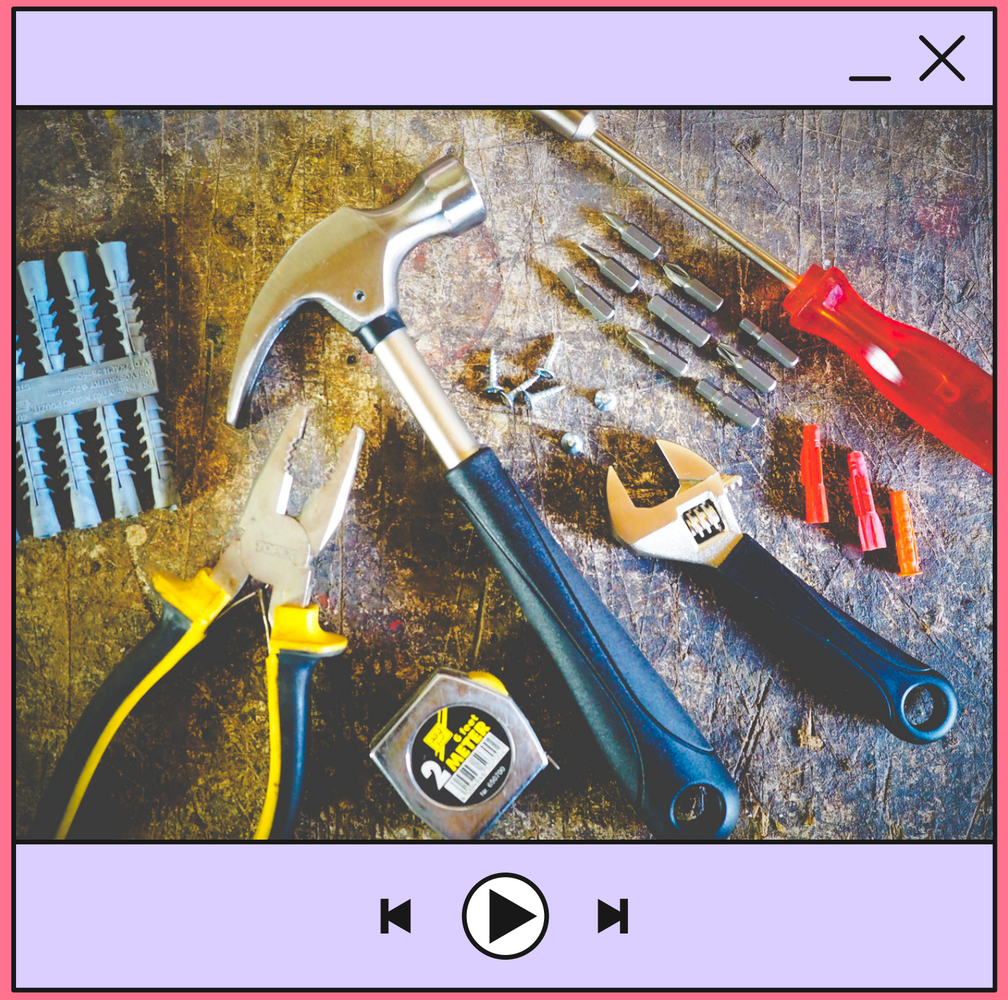
(f) Soil moisture stress conditions and supplementary irrigation for rainfed crops.

(g) Advice for irrigation timing and quantity in terms of pan evaporation.

(h) Specific information about the evolution of meteorological variables into the canopy layer in some specific cases.

(i) Microclimate inside crops in specific cases.

3. Frameworks, tools and technologies



1. Pre-processing a weather dataset from KAGGLE using Pandas
2.Training a time series forecasting model to predict temperature using Neural Prophet
3. Forecasting temperature into the future using the trained model.
4. Using Neural prophet.
5. Using pickle module.

pandas

PROPHET

What is neuralprophet

NeuralProphet bridges the gap between traditional time-series models and deep learning methods. NeuralProphet which set an industry standard for explainable, scalable, and user-friendly forecasting frameworks. With the proliferation of time series data, explainable forecasting remains a challenging task for business and operational decision making.

Presented in a user-friendly Python package, NeuralProphet uses a fusion of classic components and neural networks to produce highly accurate time series forecasts quickly. Current Prophet users will find the package to be familiar in design.
The framework provides automatic hyperparameter selection, making it a convenient and accessible tool for beginners. Advanced forecasting practitioners can incorporate domain knowledge and leverage deeper expertise with a superset of custom modules, model weight sparsification, and global modeling capabilities.
As a modular framework, NeuralProphet is composed of components that are interpretable, scalable and independently configurable. All modules are jointly trained with mini-batch stochastic gradient descent (SGD). Any model component that is trainable by SGD can be included as a module, which makes it easy to extend the framework with the state-of-the-art forecasting methods of the future.

What is pickle?

Python pickle module is used for serializing and de-serializing a Python object structure. Any object in Python can be pickled so that it can be saved on disk. What pickle does is that it “serializes” the object first before writing it to file. Pickling is a way to convert a python object (list, dict, etc.) into a character stream. The idea is that this character stream contains all the information necessary to reconstruct the object in another python script.

The data format used by pickle is Python-specific. By default, the pickle data format uses a relatively compact binary representation. If you need optimal size characteristics, you can efficiently compress pickled data.

The module pickletools contains tools for analyzing data streams generated by pickle. pickletools source code has extensive comments about opcodes used by pickle protocols.

In real world sceanario, the use pickling and unpickling are widespread as they allow us to easily transfer data from one server/system to another and then store it in a file or database. The pickle module of python contains the four methods:

1. dump(obj, file, protocol = None, * , fix_imports = True, buffer_callback = None)
2. dumps(obj, protocol = None, * , fix_imports = True, buffer_callback = None)
3. load(file, * , fix_imports = True, encoding = " ASCII ", errors = "strict ", buffers = None)
4. loads(bytes_object, * , fix_imports = True, encoding = " ASCII ", errors = " strict ", buffers = None)

The first two methods are used for the pickling process, and the next two methods are used for the unpickling process.

The difference between dump() and dumps() is that dump() creates the file which contains the serialization results, and the dumps() returns the string.

For differentiation dumps() from the dump(), the developer can remember that in the dumps() function, ' s' stands for the string.

Assumptions and constraints

- Firstly, we read the weather data from Kaggle using pandas.
- Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges
- Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits
- Then we pre-process the data and make it ready for modelling.
- Then we fix and predict a time series model with Neural Prophet.
- NeuralProphet is a python library for modeling time-series data based on neural networks. It's built on top of PyTorch and is heavily inspired by Facebook Prophet and AR-Net libraries
- Pickle in Python is primarily used in serializing and deserializing a Python object structure. In other words, it's the process of converting a Python object into a byte stream to store it in a file/database, maintain program state across sessions, or transport data over the network

Based on rainfall on the particular day we can control the drip irrigation or collect the water for the future use. Plus incase of any immediate weather change a farmer can also use methods to save his/her crop subsequently.

If we start by implementing this system in local farmer help centers so that the farmers can who doesn't have the technological support can take help of this software for their crop.

If this idea is successful then we can apply this on an app based service of linguistic language support for the farmers who has subsequent technological knowledge so that each and everyone no matter what language they speak can take help of this service.

SCALABILITY AND USABILITY

CODE IMPLEMENTATION:

```
!pip install neuralprophet
```

```
import pandas as pd
from neuralprophet import NeuralProphet
from matplotlib import pyplot as plt
import pickle
```

```
df = pd.read_csv('weather.csv')
df.head()
```

	Precipitation	Date	Month	Week of	Year	City	Code	Location	State	Temperature.Avg Temp	Temperature.Max Temp	Temperat
0	0.00	1/3/2016	1	3	2016	Birmingham	BHM	Birmingham, AL	Alabama	39	46	
1	0.00	1/3/2016	1	3	2016	Huntsville	HSV	Huntsville, AL	Alabama	39	47	
2	0.16	1/3/2016	1	3	2016	Mobile	MOB	Mobile, AL	Alabama	46	51	
3	0.00	1/3/2016	1	3	2016	Montgomery	MGM	Montgomery, AL	Alabama	45	52	
4	0.01	1/3/2016	1	3	2016	Anchorage	ANC	Anchorage,	Alaska	34	38	

CODE IMPLEMENTATION:

```
[4] df.columns
```

```
Index(['Precipitation', 'Date', 'Month', 'Week of', 'Year', 'City', 'Code',
       'Location', 'State', 'Temperature.Avg Temp', 'Temperature.Max Temp',
       'Temperature.Min Temp', 'Wind.Direction', 'Wind.Speed'],
      dtype='object')
```

```
melb = df[df['City']=='Birmingham']
melb['Date'] = pd.to_datetime(melb['Date'])
melb.head()
```

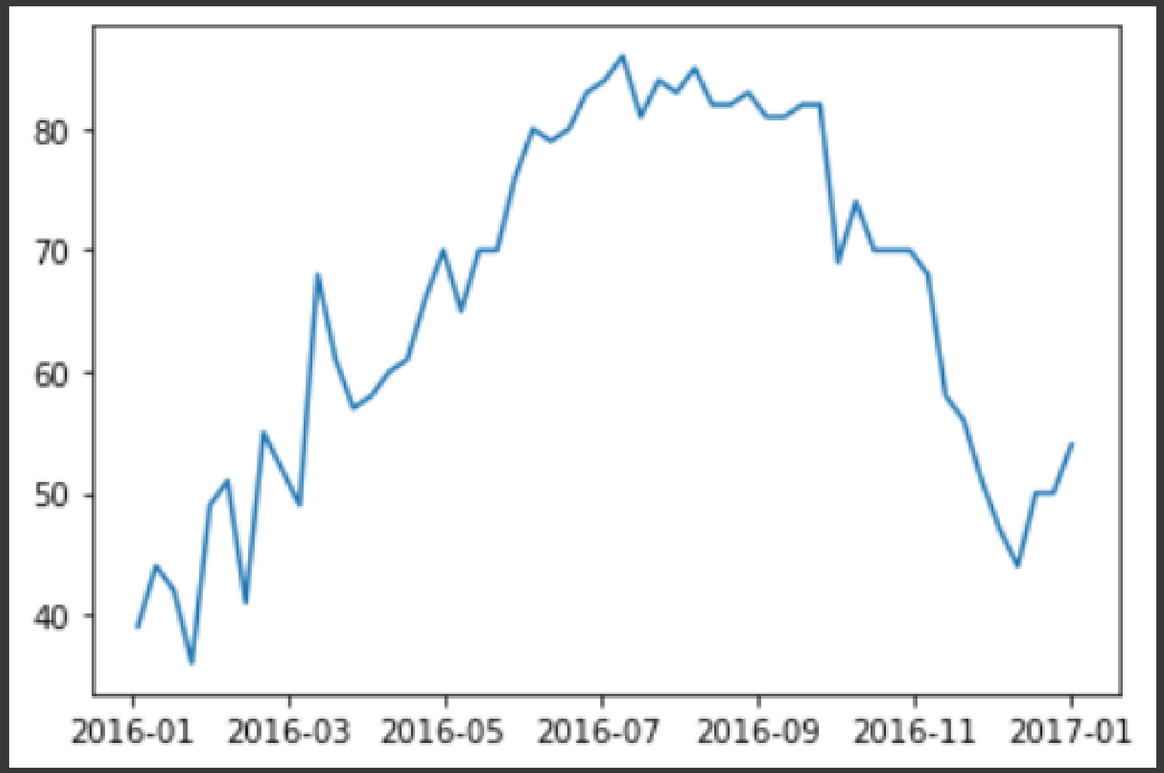
```
WARNING - (py.warnings._showwarnmsg) - /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
```

	Precipitation	Date	Month	Week of	Year	City	Code	Location	State	Temperature.Avg Temp	Temperature.Max Temp	Temperat
0	0.00	2016-01-03	1	3	2016	Birmingham	BHM	Birmingham, AL	Alabama	39	46	
315	0.52	2016-01-10	1	10	2016	Birmingham	BHM	Birmingham, AL	Alabama	44	52	
630	0.34	2016-01-17	1	17	2016	Birmingham	BHM	Birmingham, AL	Alabama	42	52	

CODE IMPLEMENTATION:

```
plt.plot(melb['Date'], melb['Temperature.Avg Temp'])
plt.show()
```



CODE IMPLEMENTATION:

```
melb['Year'] = melb['Date'].apply(lambda x: x.year)
melb = melb[melb['Year']<=2017]
plt.plot(melb['Date'], melb['Temperature.Avg Temp'])
plt.show()
```

```
WARNING - (py.warnings._showwarnmsg) - /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/ind
  """Entry point for launching an IPython kernel.
```

CODE IMPLEMENTATION:

```
data = melb[['Date', 'Temperature.Avg Temp']]
data.dropna(inplace=True)
data.columns = ['ds', 'y']
data.head()
```

```
WARNING - (py.warnings._showwarnmsg) - /usr/local/lib/python3.7/dist-packages/ipykernel_lau
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_gui
```

	ds	y
0	2016-01-03	39
315	2016-01-10	44
630	2016-01-17	42
945	2016-01-24	36
1260	2016-01-31	49

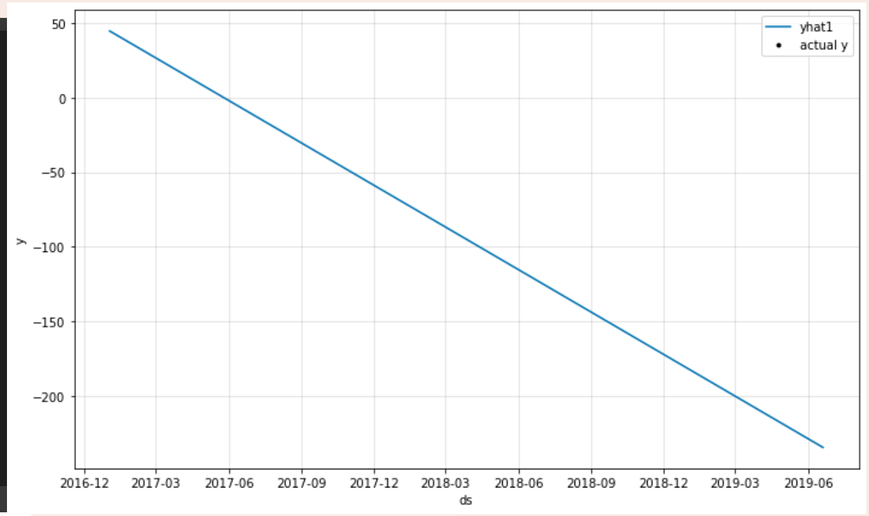
CODE IMPLEMENTATION:

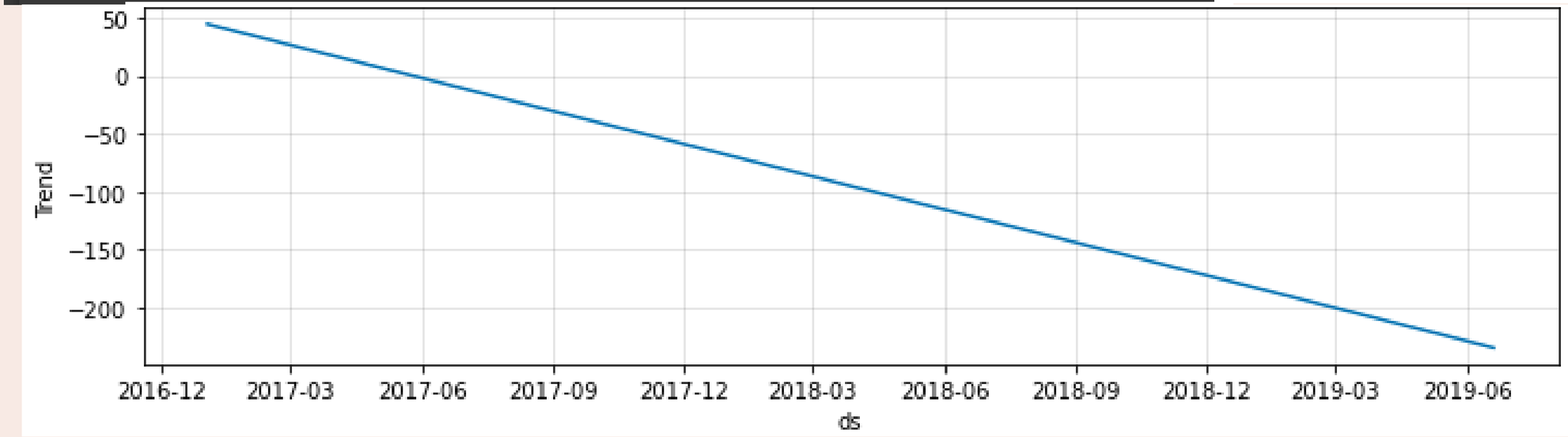
```
m = NeuralProphet()
model = m.fit(data, freq='D', epochs=1000)
```

```
INFO - (NP.utils.set_auto_seasonalities) - Disabling yearly seasonality. Run NeuralProphet with yearly_seasonality=True to over
INFO - (NP.utils.set_auto_seasonalities) - Disabling weekly seasonality. Run NeuralProphet with weekly_seasonality=True to over
INFO - (NP.utils.set_auto_seasonalities) - Disabling daily seasonality. Run NeuralProphet with daily_seasonality=True to overri
INFO - (NP.config.set_auto_batch_epoch) - Auto-set batch_size to 16
99% 188/189 [00:00<00:00, 385.18it/s]
INFO - (NP.utils_torch.lr_range_test) - lr-range-test results: steep: 5.62E-01, min: 2.87E-02
100% 189/189 [00:00<00:00, 429.06it/s]
INFO - (NP.utils_torch.lr_range_test) - lr-range-test results: steep: 5.62E-01, min: 3.57E-02
100% 189/189 [00:00<00:00, 447.05it/s]
INFO - (NP.utils_torch.lr_range_test) - lr-range-test results: steep: 5.62E-01, min: 2.87E-02
INFO - (NP.forecaster._init_train_loader) - lr-range-test selected learning rate: 5.62E-01
Epoch[1000/1000]: 100%| 1000/1000 [00:15<00:00, 62.86it/s, SmoothL1Loss=0.0033, MAE=2.91, RMSE=3.88, RegLoss=0]
```

CODE IMPLEMENTATION:

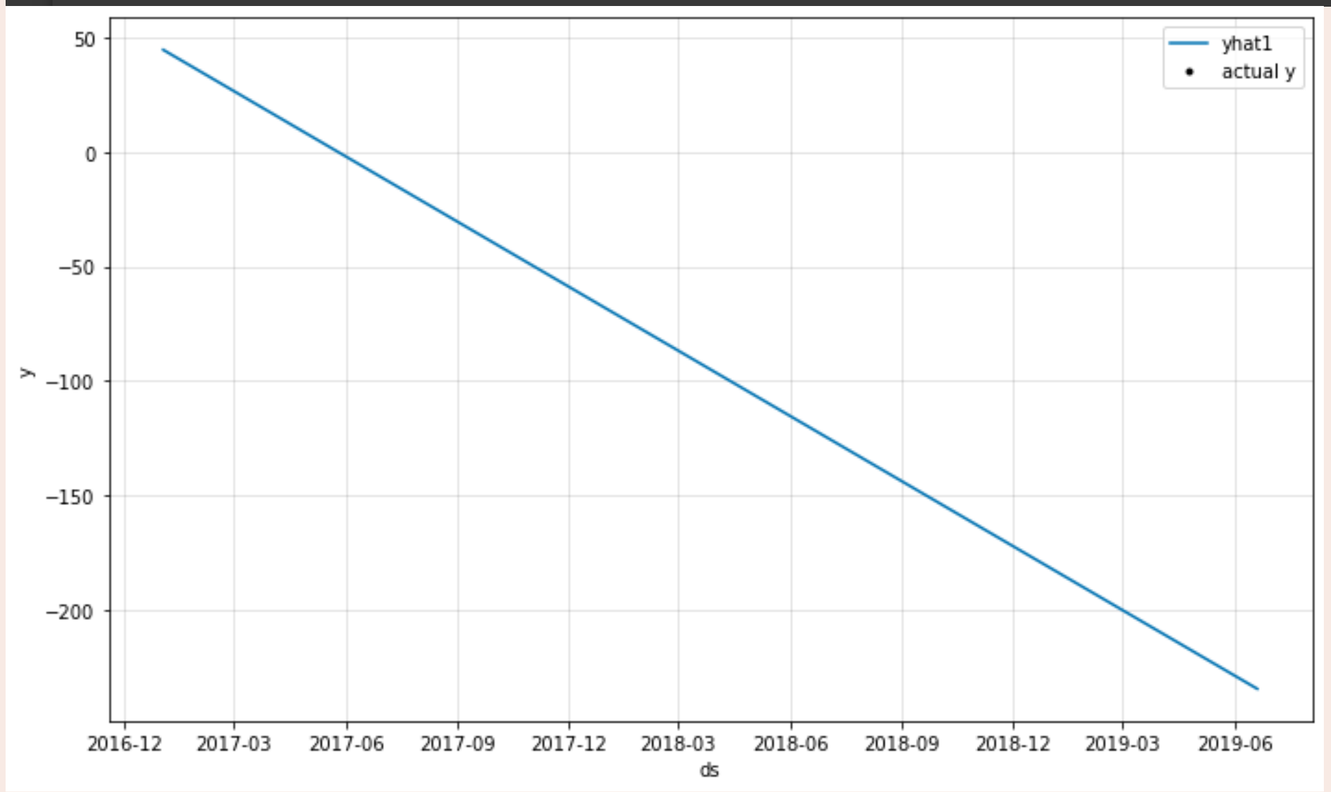
```
future = m.make_future_dataframe(data, periods=900)
forecast = m.predict(future)
forecast.head()
plot1 = m.plot(forecast)
plt2 = m.plot_components(forecast)
with open('saved_model.pkl', "wb") as f:
    pickle.dump(m, f)
```





CODE IMPLEMENTATION:

```
del m
with open('saved_model.pkl', "rb") as f:
    m = pickle.load(f)
future = m.make_future_dataframe(data, periods=900)
forecast = m.predict(future)
forecast.head()
plot1 = m.plot(forecast)
```



Find the complete code in github

Thank you for listening!