Demo_S2T_LT_Ichlasiana_Amallia

October 17, 2020

1 Speech to Text and Language Translator

IBM Watson™ Speech to Text is a cloud-native solution that uses deep-learning AI algorithms to apply knowledge about grammar, language structure, and audio/voice signal composition to create customizable speech recognition for optimal text transcription and the IBM Watson™ Language Translator allows us to translate text programmatically from one language into another language.

The goal of this project is to implement these technologies using Python.

1.0.1 References for this project

- https://github.com/watson-developer-cloud/python-sdk
- https://cloud.ibm.com/apidocs/speech-to-text?code=python
- https://cloud.ibm.com/apidocs/language-translator?code=python

1.1 Preparation

1.1.1 Install required package

`ibm-watson` is a Python client library to quickly get started with the various Watson APIs services. See more information about this package here.

`pandas` is a Python package that provides fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive. See more information about this package here.

Note:

- `ibm-watson` only support `python 3.5` or above.
- `ibm-watson` must be in version `4.7.1` or above.

[1]:
```
!pip install --upgrade "ibm-watson>=4.7.1"
!pip install pandas
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already up-to-date: ibm-watson>=4.7.1 in
/home/elmoallistair/.local/lib/python3.8/site-packages (4.7.1)
Requirement already satisfied, skipping upgrade: python-dateutil>=2.5.3 in
/usr/lib/python3/dist-packages (from ibm-watson>=4.7.1) (2.7.3)
Requirement already satisfied, skipping upgrade: requests<3.0,>=2.0 in
/usr/lib/python3/dist-packages (from ibm-watson>=4.7.1) (2.22.0)
Requirement already satisfied, skipping upgrade: ibm-cloud-sdk-core==1.7.3 in
```

```
/home/elmoallistair/.local/lib/python3.8/site-packages (from ibm-watson>=4.7.1)
(1.7.3)
Requirement already satisfied, skipping upgrade: websocket-client==0.48.0 in
/home/elmoallistair/.local/lib/python3.8/site-packages (from ibm-watson>=4.7.1)
(0.48.0)
Requirement already satisfied, skipping upgrade: PyJWT>=1.7.1 in
/usr/lib/python3/dist-packages (from ibm-cloud-sdk-core==1.7.3->ibm-
watson>=4.7.1) (1.7.1)
Requirement already satisfied, skipping upgrade: six in /usr/lib/python3/dist-
packages (from websocket-client==0.48.0->ibm-watson>=4.7.1) (1.14.0)
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pandas in
/home/elmoallistair/.local/lib/python3.8/site-packages (1.1.2)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/lib/python3/dist-
packages (from pandas) (2.7.3)
Requirement already satisfied: numpy>=1.15.4 in
/home/elmoallistair/.local/lib/python3.8/site-packages (from pandas) (1.19.2)
Requirement already satisfied: pytz>=2017.2 in /usr/lib/python3/dist-packages
(from pandas) (2019.3)
```

1.1.2 Import required modules

[2]:
```
from ibm_cloud_sdk_core.authenticators import IAMAuthenticator
from ibm_watson import SpeechToTextV1
from ibm_watson import LanguageTranslatorV3
from pandas import json_normalize
```

1.1.3 Specifies the sample audio

Specify the audio sample we will use, for this project we will use the file `PolynomialRegressionandPipelines.mp3`

Check out the supported audio formats here.

[3]:
```
filename = 'PolynomialRegressionandPipelines.mp3'
```

1.2 Transcribes audio to text

IBM Watson™ Speech service allows us to transcribes audio to text to enable speech transcription capabilities for applications.

See more information about this product here.

1.2.1 Add IBM Watson™ Speech to Text Credentials

We can create our instance here.

[4]:

```
API_S2T = 'b9wcelP5gQtN9F4r4fBECdF9g2vreLwo13CaU_v3AOis'
URL_S2T = 'https://api.au-syd.speech-to-text.watson.cloud.ibm.com/instances/
↪90b0c8f0-2bb1-40f3-b070-9a8b2777d20f'
```

1.2.2 Speech To Text Authentication

IBM Cloud Identity and Access Management (IAM) is the primary method to authenticate to the API.

Explanation:

- The `IAMAuthenticator` utilizes an apikey to obtain a suitable bearer token and adds it to requests with `apikey` argument.
- The `SpeechToTextV1` is the services we will use.
- The `set_the_url` will make HTTP requests with `service_url` argument.

Read more about authentication here.

[5]:
```
s2t_auth = IAMAuthenticator(API_S2T)
speech_to_text = SpeechToTextV1(authenticator=s2t_auth)
speech_to_text.set_service_url(URL_S2T)
```

1.2.3 Recognize the audio

Here we use `recognize()` method to sends audio and returns transcription results for a recognition request.

We can pass a maximum of 100 MB and a minimum of 100 bytes of audio with a request.

[6]:
```
with open(filename, mode="rb") as wav:
    s2t_response = speech_to_text.recognize(audio=wav, content_type='audio/mp3')
```

1.2.4 Explore the transcription result

The output looks like this:

```
{
    'result_index': 0,
    'results': [
        {'final': True, 'alternatives': [{'transcript': ...., 'confidence': ...}]},
        {'final': True, 'alternatives': [{'transcript': ...., 'confidence': ...}]}
            .... ,
    ]
}
```

Explanation:

- The `result_index` field provides a unique identifier for the results.
- The `results` field provides an array of information about the transcription results.
- The `final` field has a value of `true` to indicate that these results will not change, `false` for interim results, which are subject to change.

- The `alternatives` field provides an array of transcription results. For this request, the array includes a single element.
- The `confidence` field is a score that indicates the service's confidence in the transcript.
- The `transcript` field provides the results of the transcription.

Learn more about recognition result here.

[7]:
```
s2t_response.result
```

[7]:

```
{'result_index': 0,
 'results': [{'final': True,
   'alternatives': [{'transcript': 'in this video we will cover polynomial
regression and pipelines ',
     'confidence': 0.94}]},
  {'final': True,
   'alternatives': [{'transcript': "what do we do when a linear model is not the
best fit for our data let's look into another type of regression model the
polynomial regression we transform our data into a polynomial then use linear
regression to fit the parameters that we will discuss pipelines pipelines are
way to simplify your code ",
     'confidence': 0.9}]},
  {'final': True,
   'alternatives': [{'transcript': "polynomial regression is a special case of
the general linear regression this method is beneficial for describing
curvilinear relationships what is a curvilinear relationship it's what you get
by squaring or setting higher order terms of the predictor variables in the
model transforming the data the model can be quadratic which means the predictor
variable in the model is squared we use a bracket to indicated as an exponent
this is the second order polynomial regression with a figure representing the
function ",
     'confidence': 0.95}]},
  {'final': True,
   'alternatives': [{'transcript': 'the model can be cubic which means the
predictor variable is cute this is the third order polynomial regression we see
by examining the figure that the function has more variation ',
     'confidence': 0.95}]},
  {'final': True,
   'alternatives': [{'transcript': "there also exists higher order polynomial
regressions when a good fit hasn't been achieved by second or third order we can
see in figures how much the graphs change when we change the order of the
polynomial regression the degree of the regression makes a big difference and
can result in a better fit if you pick the right value in all cases the
relationship between the variable in the parameter is always linear ",
     'confidence': 0.91}]},
  {'final': True,
   'alternatives': [{'transcript': "let's look at an example from our data we
generate a polynomial regression model ",
     'confidence': 0.89}]},
```

```
  {'final': True,
   'alternatives': [{'transcript': 'in python we do this by using the poly fit
function in this example we develop a third order polynomial regression model
base we can print out the model symbolic form for the model is given by the
following expression ',
     'confidence': 0.92}]},
  {'final': True,
   'alternatives': [{'transcript': "negative one point five five seven X. one
cute plus two hundred four point eight X. one squared plus eight thousand nine
hundred sixty five X. one plus one point three seven times ten to the power of
five we can also have multi dimensional polynomial linear regression the
expression can get complicated here are just some of the terms for two
dimensional second order polynomial none pies poly fit function cannot perform
this type of regression we use the preprocessing librarian scikit learn to
create a polynomial feature object the constructor takes the degree of the
polynomial as a parameter then we transform the features into a polynomial
feature with the fit underscore transform method let's do a more intuitive
example ",
     'confidence': 0.9}]},
  {'final': True,
   'alternatives': [{'transcript': 'consider the feature shown here applying the
method we transform the data we now have a new set of features that are
transformed version of our original features as that I mention of the data gets
larger we may want to normalize multiple features as scikit learn instead we can
use the preprocessing module to simplify many tasks for example we can
standardize each feature simultaneously we import standard scaler we train the
object fit the scale object then transform the data into a new data frame on a
rate X. underscore scale there are more normalization methods available in the
pre processing library as well as other transformations we can simplify our code
by using a pipeline library there are many steps to getting a prediction for
example normalization polynomial transform and linear regression we simplify the
process using a pipeline ',
     'confidence': 0.9}]},
  {'final': True,
   'alternatives': [{'transcript': 'pipeline sequentially perform a series of
transformations the last step carries out a prediction first we import all the
modules we need then we import the library pipeline we create a list of topples
the first element in the topple contains the name of the estimator model the
second element contains model constructor we input the list in the pipeline
constructor we now have a pipeline object we can train the pipeline by applying
the train method to the pipeline object we can also produce a prediction as well
',
     'confidence': 0.89}]},
  {'final': True,
   'alternatives': [{'transcript': 'the method normalizes the data performs a
polynomial transform then outputs a prediction ',
     'confidence': 0.89}]}]}
```

1.2.5 Normalize the result

Then we normalizing `alternatives` table.

[8]:
```
json_normalize(s2t_response.result['results'],"alternatives")
```

[8]:

	transcript	confidence
0	in this video we will cover polynomial regress…	0.94
1	what do we do when a linear model is not the b…	0.90
2	polynomial regression is a special case of the…	0.95
3	the model can be cubic which means the predict…	0.95
4	there also exists higher order polynomial regr…	0.91
5	let's look at an example from our data we gene…	0.89
6	in python we do this by using the poly fit fun…	0.92
7	negative one point five five seven X. one cute…	0.90
8	consider the feature shown here applying the m…	0.90
9	pipeline sequentially perform a series of tran…	0.89
10	the method normalizes the data performs a poly…	0.89

1.2.6 Cleaning the result

We will collect the `transcript` value and save it to the `list`, each element will represent `transcript` value.

```
s2t_response.result['results']  <--- s2t_response contains two key: result_index and results.
  results
      alternatives[0]         <--- alternatives contains an one element list of dictionary.
          transcript
```

[9]:
```
s2t_responses_list = []
for responses in s2t_response.result['results']:
    s2t_responses_list.append(responses['alternatives'][0]['transcript'])

s2t_responses_list
```

[9]:
```
['in this video we will cover polynomial regression and pipelines ',
 "what do we do when a linear model is not the best fit for our data let's look
into another type of regression model the polynomial regression we transform our
data into a polynomial then use linear regression to fit the parameters that we
will discuss pipelines pipelines are way to simplify your code ",
 "polynomial regression is a special case of the general linear regression this
method is beneficial for describing curvilinear relationships what is a
curvilinear relationship it's what you get by squaring or setting higher order
terms of the predictor variables in the model transforming the data the model
can be quadratic which means the predictor variable in the model is squared we
use a bracket to indicated as an exponent this is the second order polynomial
regression with a figure representing the function ",
 'the model can be cubic which means the predictor variable is cute this is the
third order polynomial regression we see by examining the figure that the
```

```
function has more variation ',
 "there also exists higher order polynomial regressions when a good fit hasn't
been achieved by second or third order we can see in figures how much the graphs
change when we change the order of the polynomial regression the degree of the
regression makes a big difference and can result in a better fit if you pick the
right value in all cases the relationship between the variable in the parameter
is always linear ",
 "let's look at an example from our data we generate a polynomial regression
model ",
 'in python we do this by using the poly fit function in this example we develop
a third order polynomial regression model base we can print out the model
symbolic form for the model is given by the following expression ',
 "negative one point five five seven X. one cute plus two hundred four point
eight X. one squared plus eight thousand nine hundred sixty five X. one plus one
point three seven times ten to the power of five we can also have multi
dimensional polynomial linear regression the expression can get complicated here
are just some of the terms for two dimensional second order polynomial none pies
poly fit function cannot perform this type of regression we use the
preprocessing librarian scikit learn to create a polynomial feature object the
constructor takes the degree of the polynomial as a parameter then we transform
the features into a polynomial feature with the fit underscore transform method
let's do a more intuitive example ",
 'consider the feature shown here applying the method we transform the data we
now have a new set of features that are transformed version of our original
features as that I mention of the data gets larger we may want to normalize
multiple features as scikit learn instead we can use the preprocessing module to
simplify many tasks for example we can standardize each feature simultaneously
we import standard scaler we train the object fit the scale object then
transform the data into a new data frame on a rate X. underscore scale there are
more normalization methods available in the pre processing library as well as
other transformations we can simplify our code by using a pipeline library there
are many steps to getting a prediction for example normalization polynomial
transform and linear regression we simplify the process using a pipeline ',
 'pipeline sequentially perform a series of transformations the last step
carries out a prediction first we import all the modules we need then we import
the library pipeline we create a list of topples the first element in the topple
contains the name of the estimator model the second element contains model
constructor we input the list in the pipeline constructor we now have a pipeline
object we can train the pipeline by applying the train method to the pipeline
object we can also produce a prediction as well ',
 'the method normalizes the data performs a polynomial transform then outputs a
prediction ']
```

1.2.7 Final result

Then we create one single `string` from `recognized_test`.

[10]:
```python
final_result_s2t = ' '.join(s2t_responses_list)
final_result_s2t
```

[10]:
```
"in this video we will cover polynomial regression and pipelines  what do we do
when a linear model is not the best fit for our data let's look into another
type of regression model the polynomial regression we transform our data into a
polynomial then use linear regression to fit the parameters that we will discuss
pipelines pipelines are way to simplify your code  polynomial regression is a
special case of the general linear regression this method is beneficial for
describing curvilinear relationships what is a curvilinear relationship it's
what you get by squaring or setting higher order terms of the predictor
variables in the model transforming the data the model can be quadratic which
means the predictor variable in the model is squared we use a bracket to
indicated as an exponent this is the second order polynomial regression with a
figure representing the function  the model can be cubic which means the
predictor variable is cute this is the third order polynomial regression we see
by examining the figure that the function has more variation  there also exists
higher order polynomial regressions when a good fit hasn't been achieved by
second or third order we can see in figures how much the graphs change when we
change the order of the polynomial regression the degree of the regression makes
a big difference and can result in a better fit if you pick the right value in
all cases the relationship between the variable in the parameter is always
linear  let's look at an example from our data we generate a polynomial
regression model  in python we do this by using the poly fit function in this
example we develop a third order polynomial regression model base we can print
out the model symbolic form for the model is given by the following expression
negative one point five five seven X. one cute plus two hundred four point eight
X. one squared plus eight thousand nine hundred sixty five X. one plus one point
three seven times ten to the power of five we can also have multi dimensional
polynomial linear regression the expression can get complicated here are just
some of the terms for two dimensional second order polynomial none pies poly fit
function cannot perform this type of regression we use the preprocessing
librarian scikit learn to create a polynomial feature object the constructor
takes the degree of the polynomial as a parameter then we transform the features
into a polynomial feature with the fit underscore transform method let's do a
more intuitive example  consider the feature shown here applying the method we
transform the data we now have a new set of features that are transformed
version of our original features as that I mention of the data gets larger we
may want to normalize multiple features as scikit learn instead we can use the
preprocessing module to simplify many tasks for example we can standardize each
feature simultaneously we import standard scaler we train the object fit the
scale object then transform the data into a new data frame on a rate X.
underscore scale there are more normalization methods available in the pre
processing library as well as other transformations we can simplify our code by
using a pipeline library there are many steps to getting a prediction for
example normalization polynomial transform and linear regression we simplify the
process using a pipeline  pipeline sequentially perform a series of
```

```
transformations the last step carries out a prediction first we import all the
modules we need then we import the library pipeline we create a list of topples
the first element in the topple contains the name of the estimator model the
second element contains model constructor we input the list in the pipeline
constructor we now have a pipeline object we can train the pipeline by applying
the train method to the pipeline object we can also produce a prediction as well
the method normalizes the data performs a polynomial transform then outputs a
prediction "
```

1.3 Translate the text to another language

IBM Watson™ Language Translator allows us to translate text programmatically from one language into another language.

See more information about this product here.

1.3.1 Add IBM Watson™ Language Translator Credentials

We can create our instance here.

In this example, we use Language Translator Version 2018-05-01.

See about versioning here.

[11]:
```
API_LT = 'wG_Z6raX83CeLUfG_1kXEY1J62MGmph9biaSwGVWgsIJ'
URL_LT = 'https://api.au-syd.language-translator.watson.cloud.ibm.com/instances/
↪5011afb7-e529-43f5-8602-7f87785fe87b'
VER_LT = '2018-05-01'
```

1.3.2 Language Translator Authentication

IBM Cloud Identity and Access Management (IAM) is the primary method to authenticate to the API.

Explanation:

- The `IAMAuthenticator` utilizes an apikey to obtain a suitable bearer token and adds it to requests with `apikey` argument.
- The `LanguageTranslatorV3` is the services we will use.
- The `set_the_url` will make HTTP requests with `service_url` argument.

Read more about to authentication here.

[12]:
```
lt_auth = IAMAuthenticator(API_LT)
language_translator = LanguageTranslatorV3(version=VER_LT,␣
↪authenticator=lt_auth)
language_translator.set_service_url(URL_LT)
```

1.3.3 Get a list of supported languages

The `list_identifiable_languages()` method returns the language code (for example, `en` for English or `es` for Spanish) and the name of each language.

You also can see supported languages here.

[13]:
```
json_normalize(language_translator.list_identifiable_languages().get_result(),␣
↪"languages")
```

[13]:
```
   language                  name
0        af             Afrikaans
1        ar                Arabic
2        az           Azerbaijani
3        ba               Bashkir
4        be            Belarusian
..      …                     …
71       uk             Ukrainian
72       ur                  Urdu
73       vi            Vietnamese
74       zh    Simplified Chinese
75    zh-TW   Traditional Chinese

[76 rows x 2 columns]
```

1.3.4 Translate from EN to ID

The `translate()` method will translates the input text from the source language to the target language.

The `text` parameter take text in UTF-8 encoding with maximum of 50 KB (51,200 bytes) of text with a single request. In this example we use text from `final_result_s2t`.

We can specify `model_id` using format `source-target`. For example, `en-de` selects the IBM-provided base model for English-to-German translation.

Read more about this here.

[14]:
```
tl_response = language_translator.translate(text=final_result_s2t,␣
↪model_id='en-id')
tl_result = tl_response.get_result()
```

1.3.5 Explore the translation result

The output looks like this:

```
{
    'translations': [{'translation': ...}],
    'word_count': ...,
    'character_count': ...
}
```

Explanation:

- `word_count`: Number of words in the input text.
- `character_count`: Number of characters in the input text.
- `translations`: List of translation output in UTF-8, corresponding to the input text entries.

Read more about the response here.

[15]:
```
tl_result
```

[15]:

```
{'translations': [{'translation': 'dalam video ini kita akan menutupi regresi
polinomial dan jaringan pipa apa yang kita lakukan ketika model linear tidak
cocok untuk data kita mari kita lihat ke dalam jenis lain dari regresi model
regresi polinomial ini kita mengubah data kita menjadi polinomial kemudian
menggunakan regresi linear untuk sesuai dengan parameter yang kita akan membahas
regresi pipa adalah cara untuk menyederhanakan regresi polinomial umum metode
ini bermanfaat untuk menggambarkan hubungan kurvilinear apa itu hubungan
kurvilinear Ini adalah apa yang Anda dapatkan dengan menyia-nyiakan atau
mengatur urutan urutan yang lebih tinggi dari variabel prediktor dalam model
mengubah data model dapat kuadrat yang berarti variabel prediktor dalam model
adalah kuadrat kita menggunakan bracket untuk diindikasikan sebagai eksponen ini
adalah orde kedua polinomial regresi dengan angka yang mewakili fungsi model
dapat kubik lucu ini adalah orde ketiga regresi polinomial yang kita lihat
dengan memeriksa angka bahwa fungsi memiliki variasi lebih tinggi ada regresi
polinomial yang lebih tinggi ketika yang baik tidak pernah dicapai dengan urutan
kedua atau ketiga kita dapat melihat dalam angka berapa banyak grafik perubahan
ketika kita mengubah urutan regresi polinomial derajat regresi membuat perbedaan
besar dan dapat menghasilkan nilai yang lebih baik jika Anda memilih nilai yang
tepat dalam semua kasus hubungan antara variabel dalam parameter selalu linear
mari kita lihat contoh dari data kita kita menghasilkan model regresi polinomial
di python kita melakukan ini dengan menggunakan fungsi polinomial regresi dalam
contoh ini kita dapat mencetak ketiga polinomial regresi model dasar kita dapat
mencetak model bentuk simbolik untuk model diberikan oleh ekspresi berikut
negatif satu poin lima lima tujuh X. satu lucu ditambah dua ratus empat titik
delapan X. satu kuadrat ditambah delapan ribu sembilan ratus enam puluh lima X.
satu plus satu titik tiga tujuh kali sepuluh ke kekuatan lima kita juga dapat
memiliki multi dimensi polinomial linear regresi ekspresi dapat mendapatkan
rumit di sini adalah hanya beberapa istilah untuk dua dimensi kedua dimensi
polinomial tidak ada pi poly fit function tidak dapat melakukan ini jenis
regresi kita menggunakan preproseomial librarian scikit belajar untuk membuat
sebuah polinomial fitur yang konstruktor mengambil derajat polinomial ini
sebagai parameter maka kita mengubah fitur menjadi fitur polinomial dengan fitur
polinomial dengan underscore yang pas transformasi metode mari kita lakukan
contoh yang lebih intuitif mempertimbangkan fitur yang ditampilkan di sini
menerapkan metode kita mengubah data kita sekarang memiliki seperangkat fitur
baru yang diubah dari fitur asli kita sebagai yang saya sebutkan dari data yang
lebih besar kita mungkin ingin menormalkan beberapa fitur sebagai scikit belajar
sebaliknya kita dapat menggunakan modul preproseit untuk menyederhanakan banyak
```

```
tugas secara bersamaan kita impor skala standar kita melatih objek sesuai dengan
objek skala kemudian mengubah data menjadi bingkai data baru pada tingkat X.
skala kecil ada lebih banyak normalisasi metode yang tersedia di perpustakaan
pre-processing serta transformasi lainnya kita dapat menyederhanakan kode kita
dengan menggunakan pustaka pipeline ada banyak langkah untuk mendapatkan
prediksi misalnya normalisasi polinomial transform dan linear regresi kita
menyederhanakan proses menggunakan pipa pipa secara berurutan melakukan
serangkaian transformasi langkah terakhir membawa keluar prediksi terlebih
dahulu kita impor semua modul yang kita butuhkan maka kita impor pipa
perpustakaan kita membuat daftar dari topples elemen pertama dalam topple berisi
nama estimasi atau model elemen kedua berisi model konstruktor. kita masukan
daftar dalam konstruktor pipa kita sekarang memiliki objek pipa kita dapat
melatih pipa dengan menerapkan metode kereta api ke objek pipa kita juga dapat
menghasilkan prediksi juga metode menormalkan data melakukan transformasi
polinomial kemudian keluar menempatkan prediksi. '}],
 'word_count': 680,
 'character_count': 3970}
```

1.3.6 Get the translation result

We will get the `translation` value and save it to the `final_result_tl`.

```
tl_result
  translations[0]     <--- translations contains an one element list of dictionary.
      translation
```

[16]:
```
final_result_tl = tl_result['translations'][0]['translation']
```

1.3.7 Final result

[17]:
```
final_result_tl
```

[17]:
```
'dalam video ini kita akan menutupi regresi polinomial dan jaringan pipa apa
yang kita lakukan ketika model linear tidak cocok untuk data kita mari kita
lihat ke dalam jenis lain dari regresi model regresi polinomial ini kita
mengubah data kita menjadi polinomial kemudian menggunakan regresi linear untuk
sesuai dengan parameter yang kita akan membahas regresi pipa adalah cara untuk
menyederhanakan regresi polinomial umum metode ini bermanfaat untuk
menggambarkan hubungan kurvilinear apa itu hubungan kurvilinear Ini adalah apa
yang Anda dapatkan dengan menyia-nyiakan atau mengatur urutan urutan yang lebih
tinggi dari variabel prediktor dalam model mengubah data model dapat kuadrat
yang berarti variabel prediktor dalam model adalah kuadrat kita menggunakan
bracket untuk diindikasikan sebagai eksponen ini adalah orde kedua polinomial
regresi dengan angka yang mewakili fungsi model dapat kubik lucu ini adalah orde
ketiga regresi polinomial yang kita lihat dengan memeriksa angka bahwa fungsi
memiliki variasi lebih tinggi ada regresi polinomial yang lebih tinggi ketika
yang baik tidak pernah dicapai dengan urutan kedua atau ketiga kita dapat
melihat dalam angka berapa banyak grafik perubahan ketika kita mengubah urutan
```

regresi polinomial derajat regresi membuat perbedaan besar dan dapat menghasilkan nilai yang lebih baik jika Anda memilih nilai yang tepat dalam semua kasus hubungan antara variabel dalam parameter selalu linear mari kita lihat contoh dari data kita kita menghasilkan model regresi polinomial di python kita melakukan ini dengan menggunakan fungsi polinomial regresi dalam contoh ini kita dapat mencetak ketiga polinomial regresi model dasar kita dapat mencetak model bentuk simbolik untuk model diberikan oleh ekspresi berikut negatif satu poin lima lima tujuh X. satu lucu ditambah dua ratus empat titik delapan X. satu kuadrat ditambah delapan ribu sembilan ratus enam puluh lima X. satu plus satu titik tiga tujuh kali sepuluh ke kekuatan lima kita juga dapat memiliki multi dimensi polinomial linear regresi ekspresi dapat mendapatkan rumit di sini adalah hanya beberapa istilah untuk dua dimensi kedua dimensi polinomial tidak ada pi poly fit function tidak dapat melakukan ini jenis regresi kita menggunakan preproseomial librarian scikit belajar untuk membuat sebuah polinomial fitur yang konstruktor mengambil derajat polinomial ini sebagai parameter maka kita mengubah fitur menjadi fitur polinomial dengan fitur polinomial dengan underscore yang pas transformasi metode mari kita lakukan contoh yang lebih intuitif mempertimbangkan fitur yang ditampilkan di sini menerapkan metode kita mengubah data kita sekarang memiliki seperangkat fitur baru yang diubah dari fitur asli kita sebagai yang saya sebutkan dari data yang lebih besar kita mungkin ingin menormalkan beberapa fitur sebagai scikit belajar sebaliknya kita dapat menggunakan modul preproseit untuk menyederhanakan banyak tugas secara bersamaan kita impor skala standar kita melatih objek sesuai dengan objek skala kemudian mengubah data menjadi bingkai data baru pada tingkat X. skala kecil ada lebih banyak normalisasi metode yang tersedia di perpustakaan pre-processing serta transformasi lainnya kita dapat menyederhanakan kode kita dengan menggunakan pustaka pipeline ada banyak langkah untuk mendapatkan prediksi misalnya normalisasi polinomial transform dan linear regresi kita menyederhanakan proses menggunakan pipa pipa secara berurutan melakukan serangkaian transformasi langkah terakhir membawa keluar prediksi terlebih dahulu kita impor semua modul yang kita butuhkan maka kita impor pipa perpustakaan kita membuat daftar dari topples elemen pertama dalam topple berisi nama estimasi atau model elemen kedua berisi model konstruktor. kita masukan daftar dalam konstruktor pipa kita sekarang memiliki objek pipa kita dapat melatih pipa dengan menerapkan metode kereta api ke objek pipa kita juga dapat menghasilkan prediksi juga metode menormalkan data melakukan transformasi polinomial kemudian keluar menempatkan prediksi. '