Udacity Machine Learning Nanodegree

Capstone Project

Prediction of positive cases of COVID-19 in Peru using Time-Series Forecasting

Tania Incio

February 2021

1. Definition

1.1. Project Overview

The coronavirus COVID-19 has been classified by the World Health Organization as a public health emergency of international importance. The pandemic began in December 2019 in Wuhan, China and its spread was rapid and global, causing the deaths of thousands of people around the world.

In this scenario, we can say that the current COVID-19 pandemic is devastating, despite the wide implementation of control measures.

In an analysis of the panel of COVID-19 cases in Peru until February 26, 2021, the country's Ministry of Health had confirmed 1,308,722 cases, in addition to 45,903 deaths. The cases are distributed throughout the national territory, with a greater concentration in the capital Lima.

The data for Peru are alarming. In this sense, this project will aim to analyze and predict positive cases of COVID-19 in Peru using Machine Learning techniques such as Time-Series Forecasting.

There are some of real-world applications for this research, such as:

- Estimated population infected by Covid-19 using generalized logistic regression and optimization heuristics
- Interpretation of forecasts of new cases of covid-19
- Prediction of COVID-19 cases in Peru
- Prediction of the number of infections and deaths from COVID-19 in Mexico

The links of each project will be added at the end of this document as references.

My personal motivation for working on prediction of positive cases of COVID-19 is my current job, I work with an epidemiologist doctor I am in charge to analyze the COVID-19 data from Peru frequently so I am familiar with the data then It would like to apply my machine learning knowledge using time-series forecasting to this domain.

1.2. Problem Statement

Taking into account the situation that Peru is going through, this project will help epidemiologists to have one more tool to estimate the risk of this pandemic by predicting positive cases of covid-19 in Peru and some control measures can be taken.

To achieve the objective I will analyze the data of the Ministry of Health of Peru and use Machine Learning techniques such as Time-Series Forecast for the predictions of new cases.

1.3. Metrics

I used the metric R^2 or Coefficient of Determination. R2 is a statistical measure of the fitness of the predicted values to the actual values. It indicates how much variance is explained by the model. For example If my metric R2 value is 0.5, then the model can capture half of the observed variation.

$$\mathrm{R}^2 = \frac{\frac{1}{n}\sum_{i=1}^{N}\left(y_i - \widehat{y_i}_i\right)^2}{\frac{1}{n}\sum_{i=1}^{N}\left(y_i - y_i{'}\right)^2}$$

2. Analysis

2.1. Data Exploration

For this project I used the open data of the Peruvian Ministry of Health. These published data correspond to the total of reported cases that tested positive for COVID-19, by department, province and district. In addition, within the data set there are data that allow identifying the main characteristics of the patient such as age, sex and date of obtaining the positive result. At the level of update frequency of the report, the Open Data Portal indicates that the Ministry of Health carries out a daily update of the information

I worked with data obtained from March 6 to February 27, 2021, there can be many records in a single day, every record means a single confirmed case of COVID-19, the dataset contains 1,093,938 records.

As my prediction is the number of infections of COVID-19 per day so my labels will be the number of new cases of COVID-19 per day.

The dataset has the following inputs:

- UUID: Universally unique identifier, identifier of each record
- DEPARTAMENTO: State where the infected person lives
- PROVINCIA: Province where the infected person lives
- DISTRITO: District where the infected person lives
- METODODX: Method applied such as rapid test, molecular test and antigen test
- EDAD: Age of the infected person
- SEXO: Sex of the infected person
- FECHA_RESULTADO: Date where the result of the test of the infected person was known

I counted the number of records per day to obtain the number of infected people per date. After that I splitted the data in chronological order.

A sample of this dataset is included with the accompanying git repo and the full dataset can be downloaded from [here].

2.1.1. Review of Data

- Download of data from portal of peruvian government

```
# Download csv from MINSA
url="https://cloud.minsa.gob.pe/s/Y8w3wHsEdYQSZRp/download"
s=requests.get(url).content

data = pd.read_csv(io.StringIO(s.decode('utf-8')), sep=';', low_memory=False)
data.tail()
```

- Sample of records:

	FECHA_CORTE	UUID	DEPARTAMENTO	PROVINCIA	DISTRITO	METODODX	EDAD	SEXO	FECHA_RESULTADO
0	20210228	7320cabdc1aaca6c59014cae76a134e6	LIMA REGION	HUAROCHIRI	SAN ANTONIO	PR	41.0	FEMENINO	20200526.0
1	20210228	e81602051997ace8340bb8c18fe24c65	APURIMAC	ABANCAY	ABANCAY	PR	32.0	FEMENINO	20200425.0
2	20210228	cecdbf10074dbc011ae05b3cbd320a6f	APURIMAC	ABANCAY	ABANCAY	PR	34.0	FEMENINO	20200429.0
3	20210228	71ecb6bccb248b0bb2ac72ed51b5e979	APURIMAC	ANDAHUAYLAS	ANDAHUAYLAS	PR	40.0	FEMENINO	20200426.0
4	20210228	566af4276cbe9359abe93f9aa86396c3	APURIMAC	ABANCAY	ABANCAY	PR	40.0	FEMENINO	20200428.0

- Check the type of data and null values:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1332939 entries, 0 to 1332938
Data columns (total 9 columns):
 #   Column            Non-Null Count    Dtype
---  ------            --------------    -----
 0   FECHA_CORTE       1332939 non-null  int64
 1   UUID              1332939 non-null  object
 2   DEPARTAMENTO      1332939 non-null  object
 3   PROVINCIA         1332939 non-null  object
 4   DISTRITO          1332939 non-null  object
 5   METODODX          1332939 non-null  object
 6   EDAD              1332633 non-null  float64
 7   SEXO              1332939 non-null  object
 8   FECHA_RESULTADO   1330916 non-null  float64
dtypes: float64(2), int64(1), object(6)
memory usage: 91.5+ MB
```

```
# check number of nulls
data.isnull().sum()
```

```
FECHA_CORTE           0
UUID                  0
DEPARTAMENTO          0
PROVINCIA             0
DISTRITO              0
METODODX              0
EDAD                306
SEXO                  0
FECHA_RESULTADO    2023
dtype: int64
```

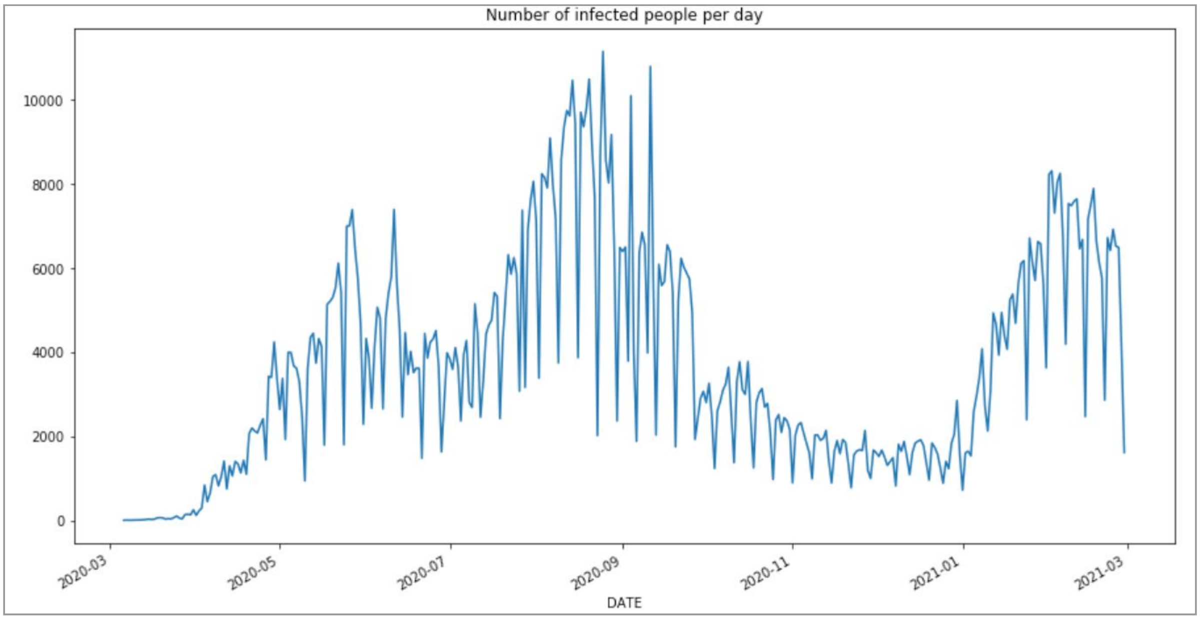
Since I will only check the FECHA_RESULTADO(date) column and add the total confirmed cases per day I will only change the data type of the FECHA_RESULTADO column to datetime and impute the null values in the preprocessing step.

2.2. Exploratory Visualization

After obtaining the number of people infected with COVID-19 per day, I plotted the confirmed cases by days, by weeks and by month in order to find any pattern of infections during this pandemic in Peru.

2.2.1. Confirmed cases per day

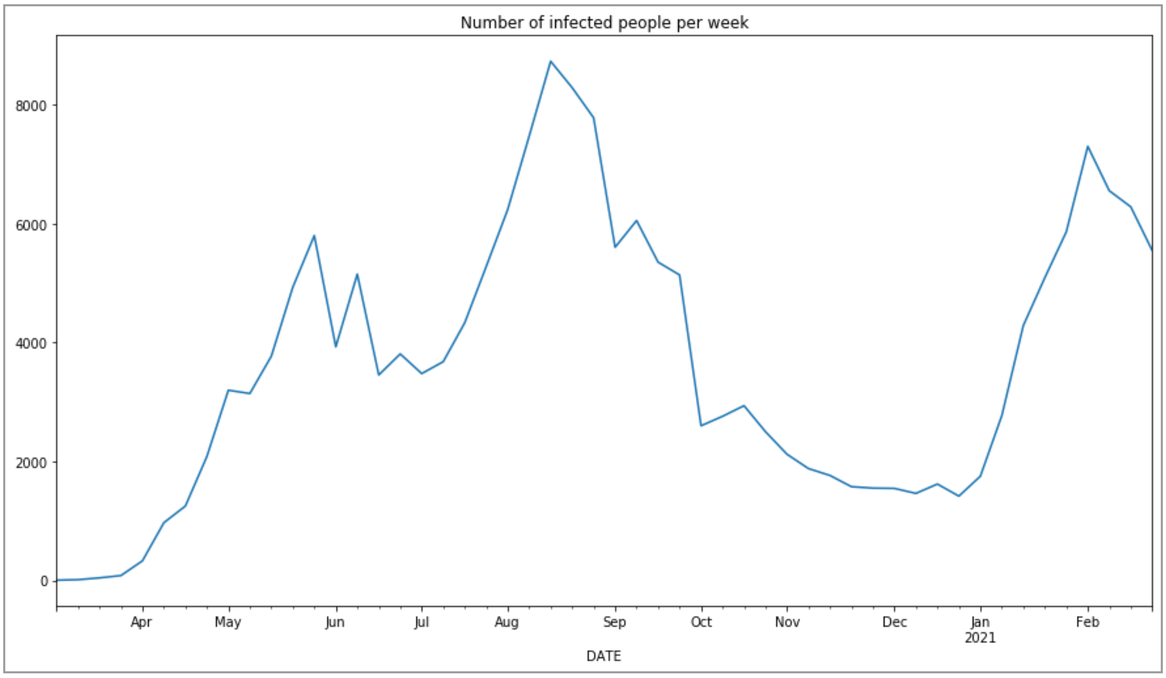
```
plt.figure(figsize=(15,8))
df_day = df_group['COUNT'].copy()
df_day.plot(title='Number of infected people per day')
plt.show()
```



We can clearly see the behavior of the coronavirus in Peru, we can visualize a first wave of infections between July and November of 2020 and the beginning of the second wave from January 1, 2021 to today

2.2.2. Confirmed cases per week

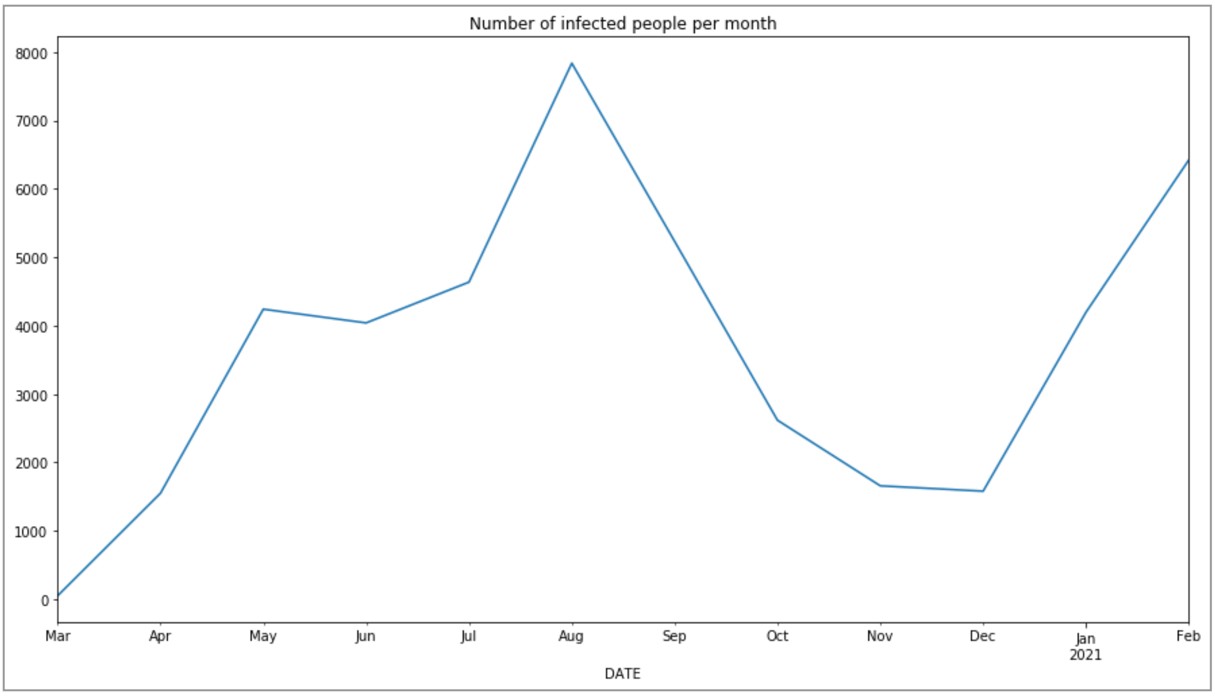
```
# resample over week (D)
freq = 'W'
# calculate the mean active power for a day
mean_week_df = df_day.resample(freq).mean()

plt.figure(figsize=(15,8))
mean_week_df.plot(title='Number of infected people per week')
plt.show();
```



In the same way we can see the two waves produced by the coronavirus in Peru if we plot the data for weeks.

2.2.3. Confirmed cases per month

```
freq = 'M'
mean_month_df = df_day.resample(freq).mean()

# display the mean values
plt.figure(figsize=(15,8))
mean_month_df.plot(title='Number of infected people per month')
plt.show();
```



After viewing the number of positive covid cases per day, weeks and months, we can find clear patterns when viewing the data for days and weeks, but having only a few days, approximately 350 days since the virus started in Peru, I decided to work the problem by grouping the cases for days and thus have more data for training.

2.3. Algorithms and Techniques

The proposed solution to this problem is to apply Machine Learning techniques such as Time-Series Forecasting to predict confirmed cases of COVID-19 in Peru.

First I will extract the data from the Open Data Portal of the Peruvian Ministry of Health. I will review the datatype of the columns, I will check the meaning of each column using the data dictionary obtained from the portal website and I will review the null data and I will make some visualizations to better understand the data.

I will apply the following techniques in these steps:

2.3.1. Preprocessing

I am gonna clean and preprocess the data of COVID-19. That COVID-19 dataset includes some data points that have missing values so in order to handle NaN rows I will drop these values specifically I am going to check the columns that add value to the project such as FECHA_RESULTADO, and some demographic characteristics of the infected people I selected the dropping technique since there a few null values and because it is more likely to create a robust model by not having any null value. At the same time I will fix the format of the FECHA_RESULTADO(date) column since that column will be important throughout the project, I will do these steps using pandas and numpy libraries.

2.3.2. Split time-series

My goal will be to take the full data from March 06, 2020 to February 27, 2021 I decided to use the full data because there are not too many data points (350 records or 350 days) then I am going to make just one single time serie for the full days I could have also decided to construct time series by years, but in 2021 we have a few data points. To complete this step I will create the function make_time_series, which will be in charge of creating time-series for training(full dataset minus the 14 days since I am going to predict the next two weeks) and testing (full dataset). The separation of the data will be chronologically.

2.3.3. Selection of the algorithm for forecasting

Before choosing the algorithm I did a research about 3 algorithms that could solve my prediction problem such as DeepAR, ARIMA and Prophet, these three algorithms have good characteristics and are used in multiple forecasting problems, I did not choose ARIMA because I read that it has difficulties to forecast changes fast in time series which would affect my problem since the confirmed cases of covid vary a lot since sometimes the state of Peru applies control measures which reduces or increases the number of cases per day significantly, I could have chosen between Prophet and DeepAR both algorithms have the ability to quickly create flexible models that solve the data variability such as positive cases, I did not choose Prophet because it can be volatile with a reduced number of observations in my case I only have 350 observations, but I have thought about applying this algorithm later, then my selection was DeepAR. DeepAR is a method based on autoregressive recurrent networks, which learns such a global model from historical data of all time series in the data set, it builds a single model for all time-series and tries to identify similarities across them.

These are the technical benefits that made me choose DeepAR:

- It can predict on items with little history items by learning from similar items, this would help my problem as my goal is to predict the next two weeks with little data.
- Variety of likelihood functions: DeepAR does not assume Gaussian noise, and likelihood functions can be adapted to the statistical properties of the data allowing for data flexibility, fits perfect for the distribution of my covid-19 data.
- Minimal Feature Engineering: The model requires minimal feature engineering, as it learns seasonal behaviour on given covariates across time series.
- It allows for simultaneous training of many related time-series, I know that in my problem I have only created one time series due to the little amount of data I have, but this model is flexible so when I have more data I could create time series per year and retrain the model.

2.3.4. Instantiating and training estimator

I am going to create a base Estimator and pass in the specific image or container that holds a 'forecasting-deepar' image model. After that I will define the estimator that will launch the training job.

Before training the model I will set some hyperparameters such as prediction_length which is the number of days that the trained model will predict, in my case it would be 14 days, the context_length which is the number of time points that the model gets to see before making a prediction it is recommendable to start with the same number as content_length, I am gonna start with that value and I could change it later, other hyperparameter is time_freq in this problem it would be 'D' for daily cases and epochs hyperparameter that is the maximum number of times to pass over the data when training it will start with the value of 50 later I will make some changes to some hyperparameters.

After that I will be ready to launch the training job.

2.3.5. Deploy the model and creating a predictor

After having trained the model, I will create an endpoint to make predictions but according to DeepAR documentation I have to do some modifications to the data inputs since the predictor expects the data in a JSON format. I will do the same when I receive the result of the prediction I will create functions to decode the data

2.3.6. Evaluating the predictor

I will make predictions and I will use the evaluation metrics such as coefficient of determination to compare the performance of the training set against the test set.

2.4. Benchmark

For the benchmark model, we will use the algorithms outlined in the paper "Prediction of the number of infections and deaths from COVID-19 in Mexico" (CONACyT-CONABIO, Inder Tecuapetla-Gómez, 2020) [1]. The paper considers the semi parametric regression model:

$$y_i = \beta_0 + \beta_1 x_i + \sum_{j=1}^{K} u_j (x_i - \tau_j)^3 + \varepsilon_i, \quad 1 \leq i \leq n$$

This model is equivalent to a mixed linear model where yi represents the number of new cases per day

Metrics	R^2
For new cases	0.9999211
For new deaths	0.9996476

3. Methodology

3.1. Data Preprocessing

Since I am familiar with the data I will make the following steps for preprocessing:

- Loading libraries

```
import pandas as pd
import numpy as np
import unicodedata
import datetime as dt
import io
import requests
import matplotlib.pyplot as plt
```

- I reviewed the number of columns and its meaning of the dataset

```
data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1332939 entries, 0 to 1332938
Data columns (total 9 columns):
 #   Column           Non-Null Count    Dtype
---  ------           --------------    -----
 0   FECHA_CORTE      1332939 non-null  int64
 1   UUID             1332939 non-null  object
 2   DEPARTAMENTO     1332939 non-null  object
 3   PROVINCIA        1332939 non-null  object
 4   DISTRITO         1332939 non-null  object
 5   METODODX         1332939 non-null  object
 6   EDAD             1332633 non-null  float64
 7   SEXO             1332939 non-null  object
 8   FECHA_RESULTADO  1330916 non-null  float64
dtypes: float64(2), int64(1), object(6)
memory usage: 91.5+ MB
```

- I removed the columns UUID since this column doesn't add any information to my problem

```
# Remove the  UUID column
data_clean = data.drop(columns=['UUID'], axis=1)
```

- I removed records with null values in the column 'EDAD' and 'FECHA_RESULTADO', since this column is important because it indicates the date of the infection.

```
# remove null values
data_clean = data_clean.dropna(subset=['FECHA_RESULTADO'])
data_clean = data_clean.dropna(subset=['EDAD'])
print(data_clean.isnull().sum())
data_clean.shape
```

```
FECHA_CORTE         0
DEPARTAMENTO        0
PROVINCIA           0
DISTRITO            0
METODODX            0
EDAD                0
SEXO                0
FECHA_RESULTADO     0
dtype: int64

(1330610, 8)
```

Of this way I no longer have any null value in my columns

- We can see that the FECHA_COLUMN format is not a proper, since it should have the date format

FECHA_RESULTADO
20200526.0
20200425.0
20200429.0
20200426.0
20200428.0

- Since the FECHA_RESULTADO column does not have a date format, we are going to format it

```
# format date
data_clean['FECHA_RESULTADO'] = data_clean['FECHA_RESULTADO'].apply(lambda x:str(x)[0:4] +
                                                                    '-' + str(x)[4:6] +
                                                                    '-' + str(x)[6:8])
```

Convert to datetime type

```
data_clean['FECHA_RESULTADO'] = pd.to_datetime(data_clean['FECHA_RESULTADO'], format='%Y-%m-%d')
data_clean.head()
```

- Rename the FECHA_RESULTADO for a english name “DATE”

```
data_clean = data_clean.rename({'FECHA_RESULTADO':'DATE'}, axis = 1)
data_clean
```

- Since we have decided to work the data grouping them by day I grouped the number of rows per day, so in this way I will get the total number of infected cases by date.

```
df_group = data_clean.groupby(['DATE']).size().reset_index(name='COUNT')
df_group = df_group.set_index('DATE')
df_group.head()
```

	COUNT
DATE	
2020-03-06	1
2020-03-07	5
2020-03-08	2
2020-03-09	3
2020-03-10	1

3.2. Implementation

3.2.1. Splitting the data chronologically

I created a function to make a time series. I used the whole dataset for the test set.

```
def make_time_series(df, freq='D', start_idx=0):
    # store time series
    time_series = []

    end_idx = len(df)
    data = df[start_idx:end_idx]

    t_start = dt.datetime.strftime(data.index[0], '%Y-%m-%d')
    t_end = dt.datetime.strftime(data.index[-1], '%Y-%m-%d')

    index = pd.date_range(start=t_start, end=t_end, freq=freq)
    time_series.append(pd.Series(data=data, index=index))


    # return list of time series
    return time_series
```

time_series variable will be used as a test set:

```
freq='D' # daily recordings

time_series = make_time_series(df_day, freq=freq)
time_series
```

```
[2020-03-06       1.0
 2020-03-07       5.0
 2020-03-08       2.0
 2020-03-09       3.0
 2020-03-10       1.0
                 ...
 2021-02-24    6928.0
 2021-02-25    6523.0
 2021-02-26    6498.0
 2021-02-27    4124.0
 2021-02-28    1615.0
 Freq: D, Name: COUNT, Length: 360, dtype: float64]
```

Function to create a training time series, I want to predict the 2 last weeks then I removed the last 14 days for the training set.

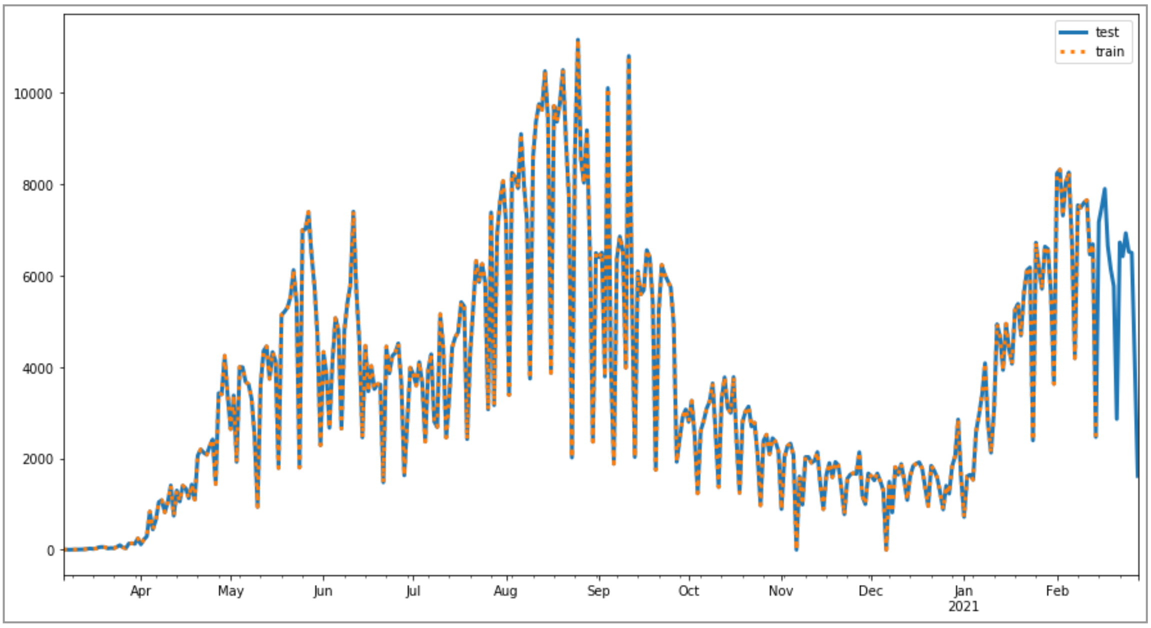
```python
# create truncated, training time series
def create_training_series(complete_time_series, prediction_length):
    # get training series
    time_series_training = []

    for ts in complete_time_series:
        time_series_training.append(ts[:-prediction_length])

    return time_series_training
```

```python
prediction_length = 14 # 2 last weeks

time_series_training = create_training_series(time_series, prediction_length)
time_series_training
```

```
[2020-03-06       1.0
 2020-03-07       5.0
 2020-03-08       2.0
 2020-03-09       3.0
 2020-03-10       1.0
                 ...
 2021-02-10    7592.0
 2021-02-11    7649.0
 2021-02-12    6462.0
 2021-02-13    6686.0
 2021-02-14    2467.0
 Freq: D, Name: COUNT, Length: 346, dtype: float64]
```

Visualization of the training and test time series

```python
time_series_idx = 0 # just we have one time-serie

plt.figure(figsize=(15,8))
time_series[time_series_idx].plot(label='test', lw=3)
time_series_training[time_series_idx].plot(label='train', ls=':', lw=3)

plt.legend()
plt.show()
```



3.2.2. Convert to JSON

Since DeepAR algorithm expects training data in a JSON format, with the fields such as **start, target.** I create a function to convert the serie to json format

```
def series_to_json_obj(ts):
    json_obj = {"start": str(ts.index[0]), "target": list(ts)}
    print(json_obj)
    return json_obj
```

3.2.3. Saving Data, Locally

I imported json library formatting data and I created a directory to save the data locally.

```
import json
import os # and os for saving

def write_json_dataset(time_series, filename):
    with open(filename, 'wb') as f:
        # for each of our times series, there is one JSON line
        for ts in time_series:
            json_line = json.dumps(series_to_json_obj(ts)) + '\n'
            json_line = json_line.encode('utf-8')
            f.write(json_line)
    print(filename + ' saved.')
```

```
# save this data to a local directory
data_dir = 'json_covid_data'

# make data dir, if it does not exist
if not os.path.exists(data_dir):
    os.makedirs(data_dir)
```

```
# directories to save train/test data
train_key = os.path.join(data_dir, 'train.json')
test_key = os.path.join(data_dir, 'test.json')

# write train/test JSON files
write_json_dataset(time_series, train_key)
write_json_dataset(time_series_training, test_key)
```

3.2.4. Uploading Data to S3

To make this data accessible to an estimator, I uploaded it to S3. I saved in the default S3 bucket.

```
sagemaker_session = sagemaker.Session()
role = get_execution_role()
bucket = sagemaker_session.default_bucket()
```

```
prefix='deepar-covid'
train_prefix   = '{}/{}'.format(prefix, 'train')
test_prefix    = '{}/{}'.format(prefix, 'test')

# uploading data to S3, and saving locations
train_path  = sagemaker_session.upload_data(train_key, bucket=bucket, key_prefix=train_prefix)
test_path   = sagemaker_session.upload_data(test_key,  bucket=bucket, key_prefix=test_prefix)
```

3.2.5. Create container image

I configured the container image to be used for the region that I am running in.

```
from sagemaker.amazon.amazon_estimator import get_image_uri
from sagemaker import image_uris

image_name = image_uris.retrieve("forecasting-deepar", boto3.Session().region_name)
```

3.2.6. Instantiate an Estimator

I define the estimator that will launch the training job

```
from sagemaker.estimator import Estimator

# dir to save model artifacts
s3_output_path = "s3://{}/{}/output".format(bucket, prefix)

# instantiate a DeepAR estimator
estimator = Estimator(sagemaker_session=sagemaker_session,
                      image_uri=image_name,
                      role=role,
                      train_instance_count=1,
                      train_instance_type='ml.c4.xlarge',
                      output_path=s3_output_path
                      )
```

3.2.7. Set a previous hyperparameters

I started with general hyperparameters such as context_length will be equal to prediction_length, time-freq will be days “D”, and epochs will be 50:

```
freq='D'
context_length=14
prediction_length = 14
hyperparameters = {
    "epochs": "50",
    "time_freq": freq,
    "prediction_length": str(prediction_length),
    "context_length": str(context_length),
    "num_cells": "50",
    "num_layers": "3",
    "mini_batch_size": "128",
    "learning_rate": "0.001",
    "early_stopping_patience": "10"
}
```

```
# set the hyperparams
estimator.set_hyperparameters(**hyperparameters)
```

3.2.8. Create a Training Job

I launched the training job, SageMaker will start an EC2 instance, download the data from S3, start training the model and save the trained model.

```
data_channels = {
    "train": train_path,
    "test": test_path
}

estimator.fit(inputs=data_channels)
```

3.2.9. Deploy and Create a Predictor

After we trained a model, we can use it to perform predictions by deploying it to a predictor endpoint.

```
from sagemaker.deserializers import JSONDeserializer
from sagemaker.serializers import IdentitySerializer
```

```
# create a predictor
predictor = estimator.deploy(
    initial_instance_count=1,
    instance_type='ml.t2.medium',
    serializer=IdentitySerializer(content_type="application/json"),
    deserializer=JSONDeserializer()
)
```

```
-----------------!CPU times: user 294 ms, sys: 16.9 ms, total: 311 ms
Wall time: 8min 33s
```

3.2.10. Create a JSON Prediction Request

DeepAR predictor expects as input data a JSON format with some parameters then we create a function to handle that:

```
def json_predictor_input(input_ts, num_samples=50, quantiles=['0.1', '0.5', '0.9']):

    instances = []
    for k in range(len(input_ts)):
        instances.append(series_to_json_obj(input_ts[k]))

    configuration = {"num_samples": num_samples,
                     "output_types": ["quantiles"],
                     "quantiles": quantiles}

    request_data = {"instances": instances,
                    "configuration": configuration}

    json_request = json.dumps(request_data).encode('utf-8')

    return json_request
```

Get predictions:

```
input_ts = time_series_training#time_series_training
target_ts = time_series# testing_series

# get formatted input time series
json_input_ts = json_predictor_input(input_ts)
json_prediction = predictor.predict(json_input_ts)

print(json_prediction)
```

3.2.11. Decoding Predictions

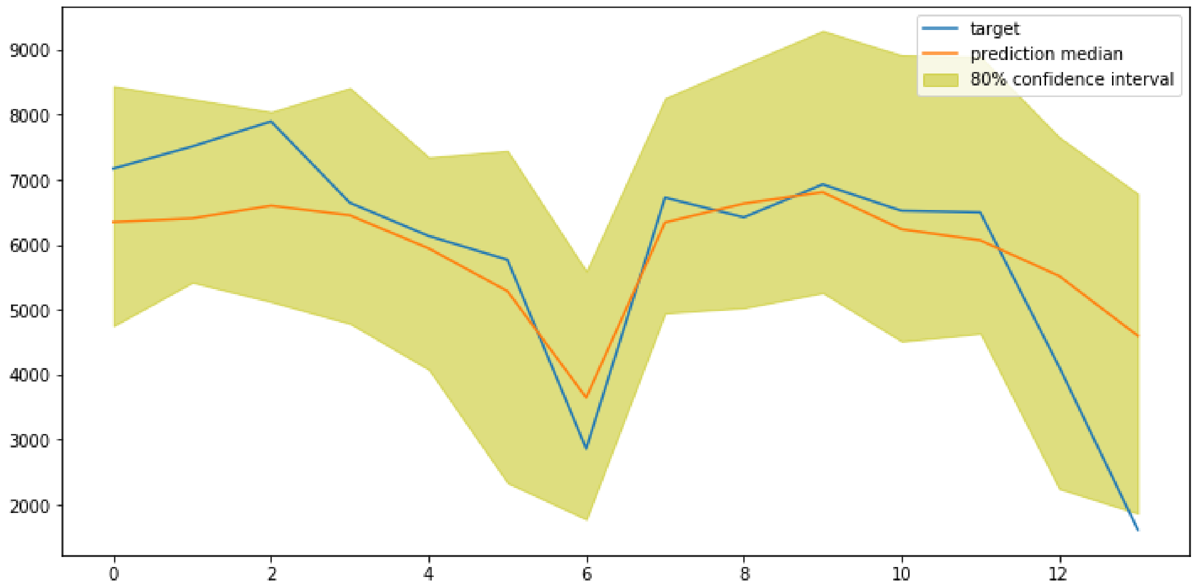
The predictor returns JSON-formatted prediction so we need to extract the predictions and data that we want to visualize. I created a function to get the quantiles:

```
def decode_prediction(prediction_data, encoding='utf-8'):
    prediction_list = []
    for k in range(len(prediction_data['predictions'])):
        prediction_list.append(pd.DataFrame(data=prediction_data['predictions'][k]['quantiles']))
    return prediction_list
```

See results of predictions:

```
# get quantiles/predictions
prediction_list = decode_prediction(json_prediction)

print(prediction_list[0])
```

	0.1	0.9	0.5
0	4753.615234	8440.416016	6349.240234
1	5422.078125	8241.997070	6405.902832
2	5123.753418	8047.131348	6601.577637
3	4788.378906	8412.750000	6453.002930
4	4081.870117	7348.327148	5944.466309
5	2338.499512	7446.619629	5286.479004
6	1776.761108	5594.989258	3646.732910
7	4950.728516	8258.001953	6343.213379
8	5030.231445	8778.042969	6634.191406
9	5261.708496	9292.825195	6809.065430
10	4516.209961	8916.866211	6238.389648
11	4635.677734	8894.954102	6067.936523
12	2249.022217	7657.090820	5520.764160
13	1867.773438	6785.363281	4598.700684



3.2.12. Check predictions and metrics

In my first attempt I got a poor metric, I will refine my hyperparameters in the next step.

```
from sklearn.metrics import r2_score
target_ts_metric = target_ts[0][-14:].values
pred_ts_metric = prediction_list[0]['0.5'].values
print(r2_score(target_ts_metric, pred_ts_metric))
```

```
0.6261934647994614
```

3.3. Refinement

I will check some hyperparameters in order to get a better metric.

My intuition is that the data set being pretty small with a rather short time-series, three layers tend to overfit more so I will decrease to two layers, also I will get a smaller learning rate in order to learn longer and I increased the number of epochs.

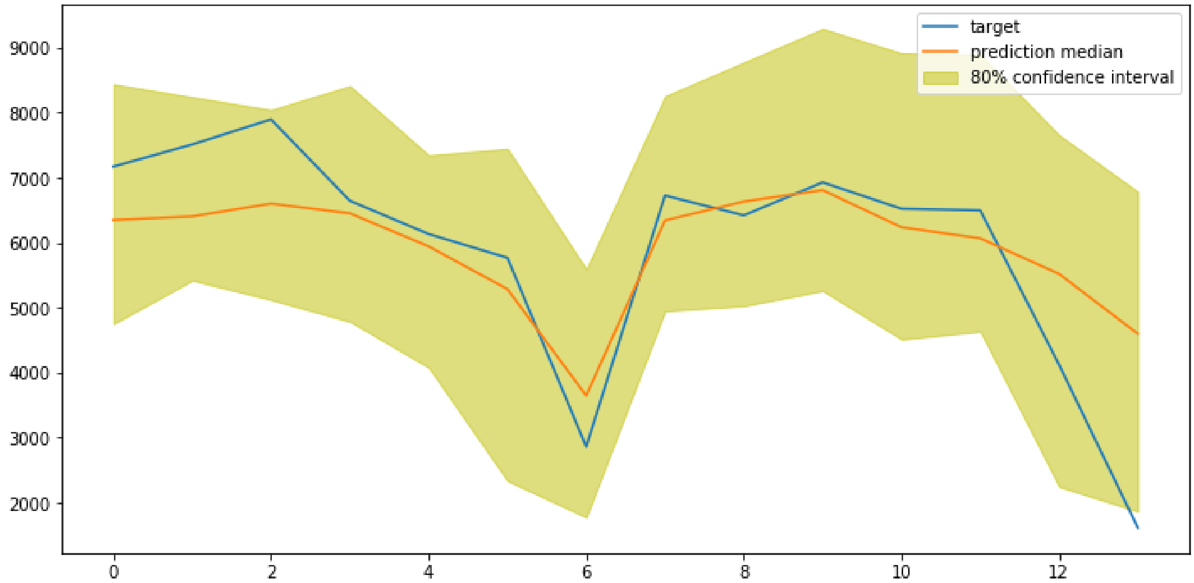
```
freq='D'
context_length=14 # same as prediction_length
prediction_length = 14
hyperparameters = {
    "epochs": "250",
    "time_freq": freq,
    "prediction_length": str(prediction_length),
    "context_length": str(context_length),
    "num_cells": "50",
    "num_layers": "2",
    "mini_batch_size": "128",
    "learning_rate": "0.0001",
    "early_stopping_patience": "10"
}
```

4. Results

4.1. Model Evaluation and Validation

After that I changed the hyperparameters and the evaluation metrics increased to 0.85448.

```
from sklearn.metrics import r2_score
target_ts_metric = target_ts[0][-14:].values
pred_ts_metric = prediction_list[0]['0.5'].values
print(r2_score(target_ts_metric, pred_ts_metric))

0.8544873557245587
```



4.2. Justification

The final model did not exceed the Benchmark model, it is very likely that it is due to the little data or so much variability of the covid data or we could do research to find better hyperparameters.

However I think it is a good score, I am going to use this model to predict the covid data for the next few weeks and will improve the algorithm over time.

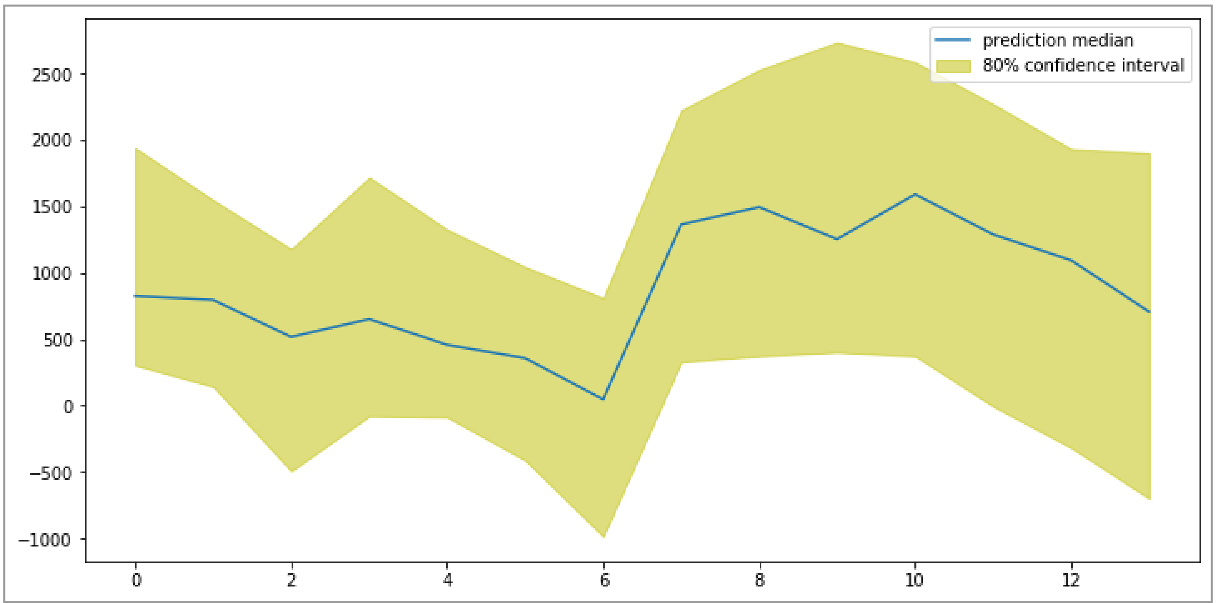
Model	R2
DeepAR	85%
BenchMark Regression model	9999211%

Predicting the Future

I would like to know how many covid-19 infections will be in Peru for the next week so the target parameters of the predictor will be an empty list because this week has no data. I started my prediction at the beginning of March, after that I get and decode the prediction response

```
start_date = '2021-03-01'
timestamp = '00:00:00'

# formatting start_date
start_time = start_date +' '+ timestamp

request_data = {"instances": [{"start": start_time, "target": []}],
                "configuration": {"num_samples": 50,
                                  "output_types": ["quantiles"],
                                  "quantiles": ['0.1', '0.5', '0.9']}
                }

json_input = json.dumps(request_data).encode('utf-8')

print('Requesting prediction for '+start_time)
```

```
# get prediction response
json_prediction = predictor.predict(json_input)

prediction_march = decode_prediction(json_prediction)
```



We can see how many confirmed cases will be in the next 14 days

```
prediction_march

[             0.1          0.9          0.5
 0    303.074799  1939.250732   824.907288
 1    142.609955  1546.614502   794.869385
 2   -492.904114  1178.477295   516.716187
 3    -79.539917  1717.063232   650.337280
 4    -87.222260  1325.477051   456.989563
 5   -412.086365  1043.897095   356.901245
 6   -985.091187   810.658447    46.637665
 7    328.371948  2223.531494  1362.629272
 8    372.151550  2527.424072  1493.013184
 9    400.483978  2733.230469  1252.137817
 10   371.970032  2584.092773  1589.176758
 11    -6.176483  2268.825928  1287.738770
 12  -318.479706  1930.701782  1092.667358
 13  -699.346802  1900.874023   706.151123]
```

Conclusion

- It was great to work with real data, especially because of a problem that all countries are experiencing. I learned that it takes a little more time to analyze the data and understand it.
- I learned to use machine learning methods like DeepAr to solve series time problems
- I will improve my techniques to tune hyperparameters and try with other machine learning models

References

https://espanol.cdc.gov/coronavirus/2019-ncov/cases-updates/forecasts-cases.html

http://www.coeeci.org.pe/wp-content/uploads/2020/03/Prediccion-Covid19-Peru-mar_24.pdf

https://www.biodiversidad.gob.mx/media/1/atlas/files/predictCOVID.pdf[1]

https://arxiv.org/pdf/2004.01207.pdf