Antonio Parolini,
Edgar Pocaterra,
Meina Bian,
Michael Adut,
Maxwell Snyder

Readme
Crypto Price Discovery Tool

CU ArgoML Project 2
Major Procedure and NN showcase case(window= 5,7,8,15,20)

Main Process

Alpacha API,

- Close price(Decide for Crypto), 92 Technical Indicators;
- https://alpaca.markets/
- API for Stock and Crypto Trading

Apply PCA for feature deduction

- identify the underlying dependencies of a dataset and to reduce its dimensionality significantly attending to them.
- This technique is beneficial for processing data sets with hundreds of variables while maintaining, at the same time, most of the information from the original data set.
- Once we have selected the principal components, the data must be projected onto them. A projection for one dimension
- The final reduced dataset will explain certain of the variance of the original one

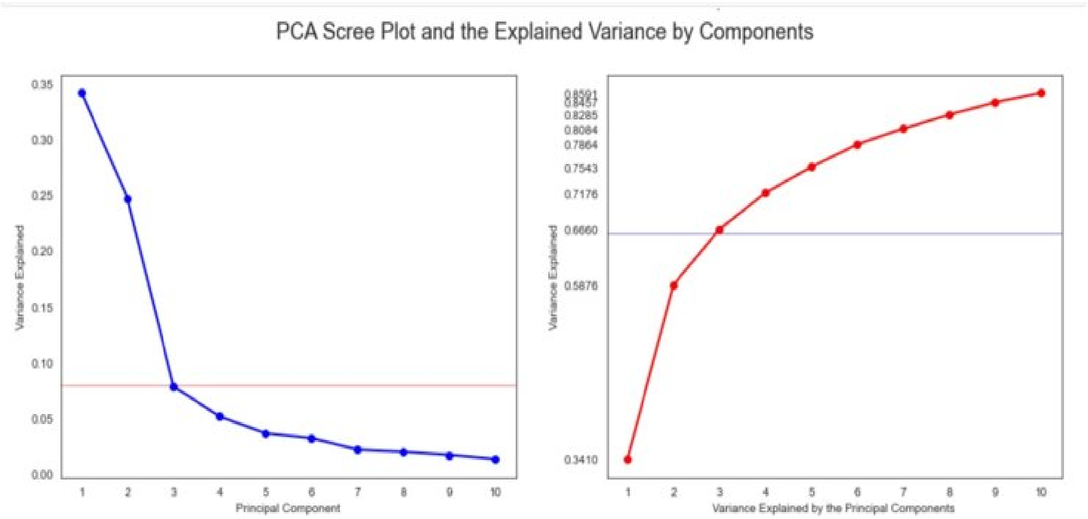
Build the Neural Network Model

- Two Hidden Layers

Process 1 PCA

- **Features dimensionality deduction through standardization.**
- **PCA analysis which suggested using K=3 for the # of primary components**
- **Trans_fit feature data matrix to 3 Features Data for building the model**

Standardize	Standardize the data (Center and scale).
Calculate	Calculate the Eigenvectors and Eigenvalues from the covariance matrix or correlation matrix (One could also use Singular Vector Decomposition).
Sort	Sort the Eigenvalues in descending order and choose the K largest Eigenvectors (Where K is the desired number of dimensions of the new feature subspace $k \le d$). (In our case K=3)
Construct	Construct the projection matrix W from the selected K Eigenvectors.
Transform	Transform the original dataset X via W to obtain a K-dimensional feature subspace Y.



Alternative to PCA
Process CNN

Applying CNNs to a univariate 1D time series Data:

- 1) Import Keras libraries and dependencies
- 2) Define a function that extracts features and outputs from the sequence.
- 3) Reshape the input X in a format that is acceptable to CNN models
- 4) Design the CNN model architecture of convolutional layers
- 5)Train the model and test it on our univariate sequence.
  - (Conv-1D),
  - pooling(max-pooling in our case),
  - flattening layer
  - fully connected neural layers.

Primary components of a Deep CNN model for time series forecasting. The primary layers of an ordinary CNN model.

- 1. Convolutional Layer
- 2. Pooling Layer
- 3. Fully Connected Layer

CNN vs RNN

- CNNs are **computationally cheaper** than RNNs: <u>CNN learns by batch</u> while RNNs train sequentially. As such, RNN can't use parallelization because it must wait for the previous computations.
- CNNs don't have the assumption that history is complete: Unlike RNNs, CNNs learn patterns within the time window. If you have **missing data, CNNs should be useful**.
- CNNs can look forward: RNN models only learn from data before the timestep it needs to predict. **CNNs (with shuffling) can see data** from a broader perspective.
- More active research in CNN: there are some arguments that **RNN / LSTM is becoming irrelevant**. Whether it's true or not, I think it depends on how we look at it.

MLP

Metric for optimality criterion

Neural Networks & Statistics,

Minimize mean squared error (MSE).

Model **Performance Tuning:**

Reduce Overfitting With Dropout

Accelerate Training With Batch Normalization

Halt Training, Early Stopping,

- monitor the loss on the training dataset a validation dataset (a subset training set, not used to fit the model).
- as loss for the validation set starts to show signs of overfitting, the training process can be stopped.

Requirement for main.v2 vast ai (GPU)version

hvplot==0.7.3
matplotlib==3.5.0
numpy==1.20.3
pandas==1.3.4
requests==2.26.0
scikit-learn==1.0.1
seaborn==0.11.2
ta==0.9.0
tensorflow==2.8.0

Requirement for main.v3 jupyter notebook version

jupyter core	: 4.6.3
jupyter-notebook	: 5.7.11
qtconsole	: 5.2.2
ipython	: 7.18.1
ipykernel	: 5.3.4
jupyter client	: 6.1.7
jupyter lab	: not installed
nbconvert	: 5.6.1
ipywidgets	: 7.6.5
nbformat	: 5.1.3
traitlets	: 5.0.5

*all module version check file "requirement_jupyter.txt" as is shown in the right

requirement_jupyter.txt - Notepad

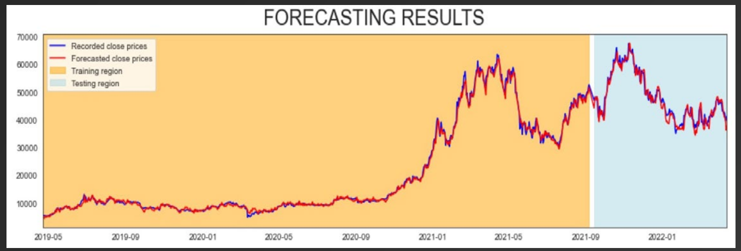
File Edit View

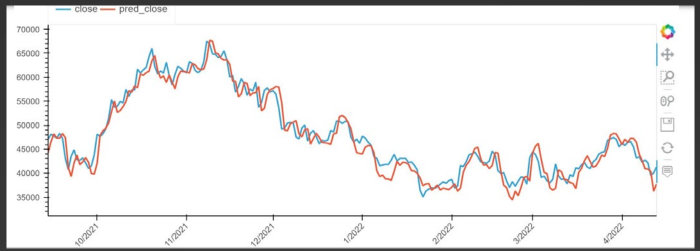
```
jupyter core     : 4.6.3
jupyter-notebook : 5.7.11
qtconsole        : 5.2.2
ipython          : 7.18.1
ipykernel        : 5.3.4
jupyter client   : 6.1.7
jupyter lab      : not installed
nbconvert        : 5.6.1
ipywidgets       : 7.6.5
nbformat         : 5.1.3
traitlets        : 5.0.5
absl-py==1.0.0
astunparse==1.6.3
async-generator==1.10
attrs @ file:///home/conda/feedstock_root/build_artifacts/attrs_1640799537051/wo
backcall==0.2.0
beautifulsoup4==4.10.0
bleach @ file:///home/conda/feedstock_root/build_artifacts/bleach_1629908509068/
bokeh @ file:///D:/bld/bokeh_1637615407028/work
Bottleneck @ file:///D:/bld/bottleneck_1636024356480/work
brotlipy @ file:///D:/bld/brotlipy_1636012439160/work
bs4==0.0.1
cached-property==1.5.2
cachetools==5.0.0
Cartopy @ file:///D:/bld/cartopy_1642061236150/work
census @ file:///home/conda/feedstock_root/build_artifacts/census_1622404746346/
certifi==2021.10.8
cffi @ file:///D:/bld/cffi_1636046306023/work
cftime @ file:///D:/bld/cftime_1642887679318/work
charset-normalizer @ file:///home/conda/feedstock_root/build_artifacts/charset-r
click @ file:///D:/bld/click_1645238363756/work
cloudpickle @ file:///home/conda/feedstock_root/build_artifacts/cloudpickle_1631
colorama @ file:///tmp/build/80754af9/colorama_1603211150991/work
colorcet==3.0.0
configparser==5.2.0
crayons==0.4.0
cryptography @ file:///D:/bld/cryptography_1639699494546/work
cycler @ file:///home/conda/feedstock_root/build_artifacts/cycler_1635519461629/
cytoolz==0.11.2
dask==2021.10.0
datashader==0.13.0
datashape==0.5.4
debugpy==1.5.1
decorator==5.1.1
defusedxml @ file:///home/conda/feedstock_root/build_artifacts/defusedxml_16152
distributed @ file:///C:/ci/distributed_1635950227219/work
entrypoints @ file:///home/conda/feedstock_root/build_artifacts/entrypoints_164
finta==1.3
fire @ file:///home/conda/feedstock_root/build_artifacts/fire_1611345183397/work
flatbuffers==2.0
fonttools @ file:///D:/bld/fonttools_1643722755699/work
fsspec @ file:///home/conda/feedstock_root/build_artifacts/fsspec_1645566723803/
future @ file:///D:/bld/future_1635819645981/work
gast==0.5.3
geojson==2.5.0
geopandas @ file:///home/conda/feedstock_root/build_artifacts/geopandas_1638382
geoviews==1.9.4
gmaps==0.9.0
google-auth==2.6.2
google-auth-oauthlib==0.4.6
google-pasta==0.2.0
graphviz==0.19.1
greenlet==1.1.2
grpcio==1.44.0
h11==0.13.0
h5py==3.6.0
HeapDict==1.0.1
holoviews==1.14.8
```

Model Summary

Metric Mse
frequently adopted optimality criterion, in both the neural networks and the statistics communities, is minimizing the mean squared error (MSE).

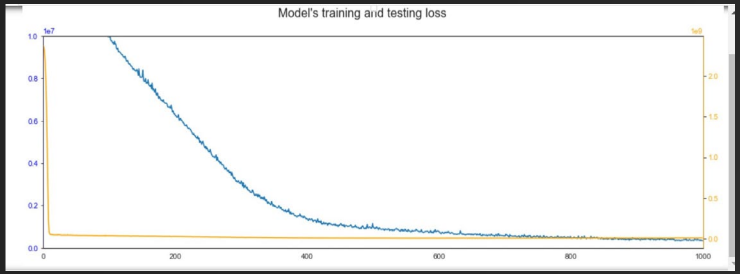
Model	PCA	NN WINDOW(step)	MSE	Overfit (Loss Figure)
nn1_5	3	5	1764.1593	No
nn1_7	3	7	1588.5481	No
nn1_8	3	8	2178.061	No
nn1_13	3	13	2159.9355	Yes(tail/verysmall)
nn1_15	3	15	3100.3516	YES(tail/small)
nn1_20	3	20	3095.867	No

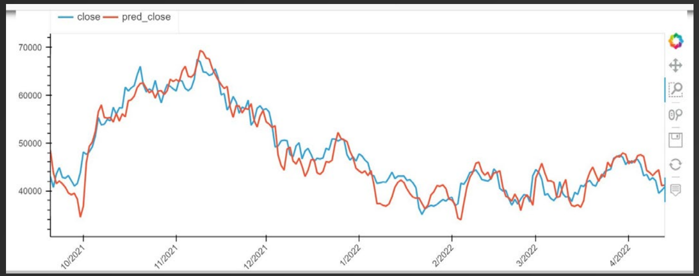




Model Results: 7-Day Window Showcase

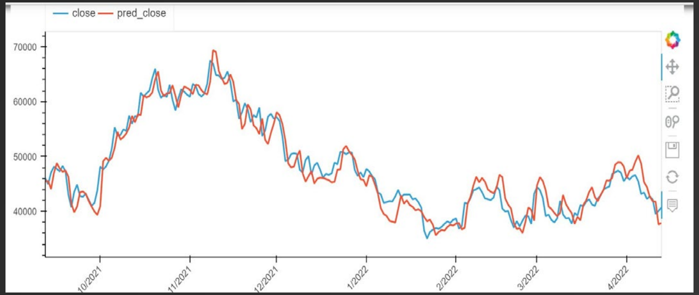
- No overfit and mse is the lowest with 1588.5481

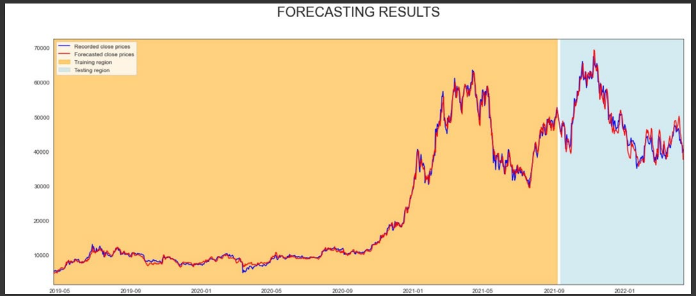




Model Results: 13-Day Window Showcase

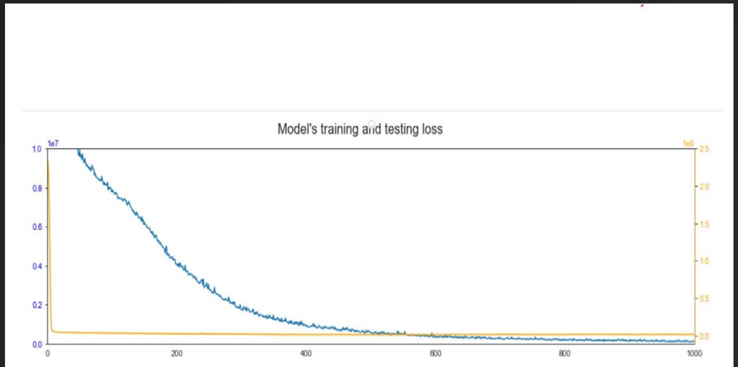
- Model start to overfit when window= 13.
- when window= 13.

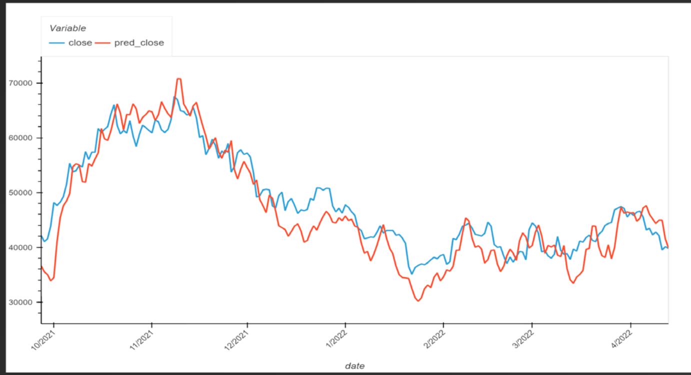


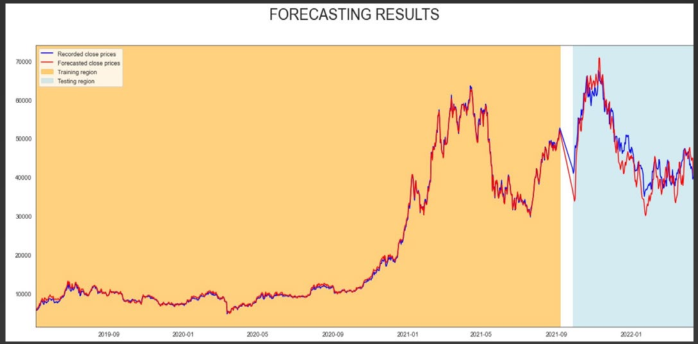


Model Results: 5-Day Window Showcase

- No overfit, but with higher mse when window= 5 compared with window=7

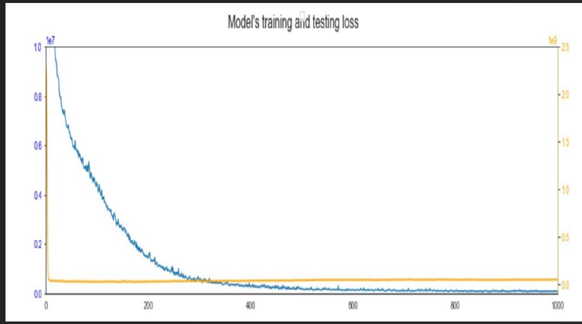


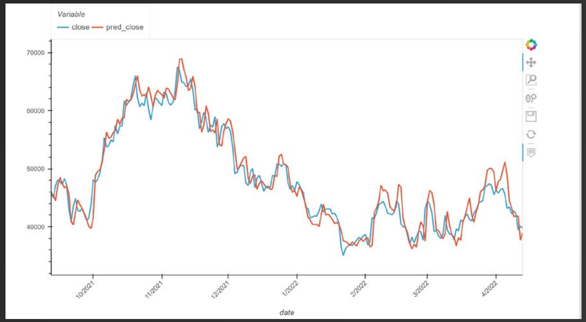




Model Results: 20-Day Window Showcase

- Overfit enlarger when window = 20





Model Results: 30-Day Window Showcase

- Overfit enlarger when window = 30