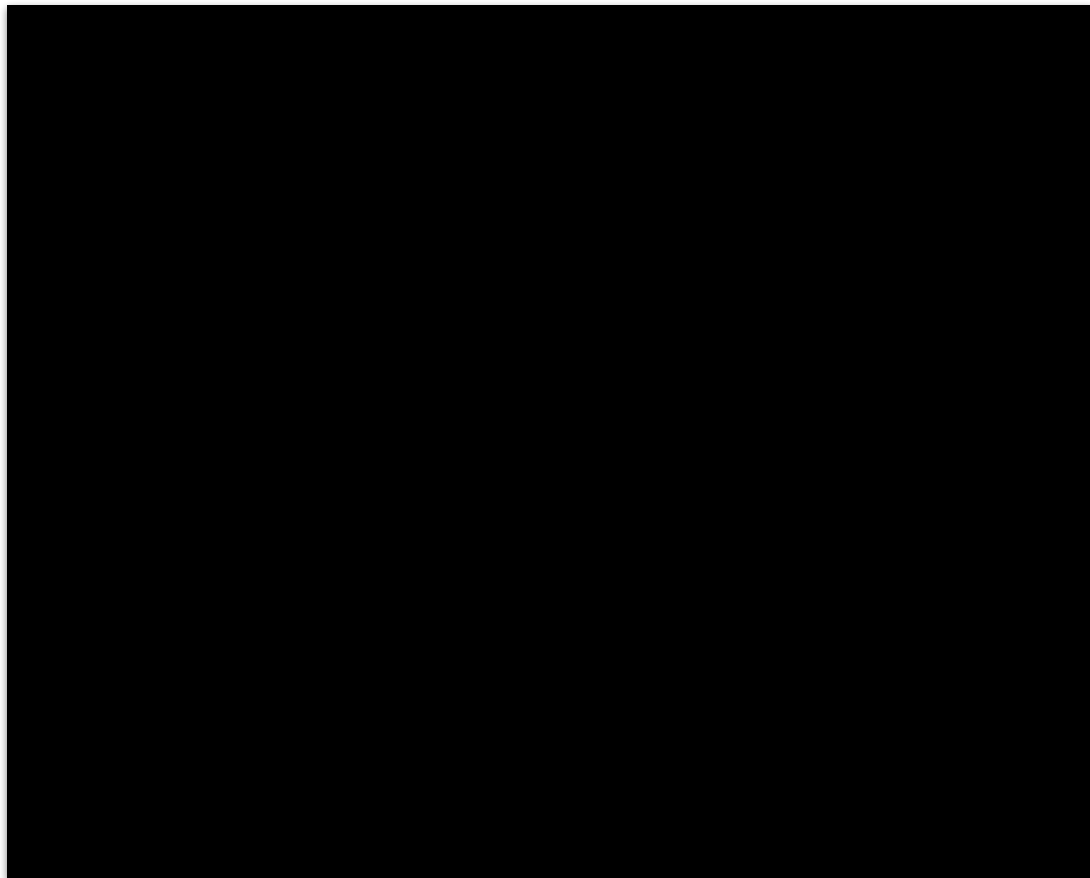


# Options Prediction with Advanced DL

Project by-  
Jacky Chow  
Jerry Huang  
Jumana Nadir



# Understanding Options Pricing



Lets watch a video first!

- Options are financial derivatives that gives the right to buy (call) or sell (put) a security at or before a certain date.
- We use deep learning to construct models that try to price.
- Options using historical data.



## 5 Basic Characteristics of Every Option

- Underlying asset
- Call vs. put
- Strike price
- Expiration date
- American vs. European

The first three functions: used to calculate/measure the actual value of options;

The last two functions: used to calculate/measure the time value of options.



# Why ?

- The value of a European option can be modeled with the commonly-used

$$C = N(d_1)S_t - N(d_2)Ke^{-rt}$$

$$\text{where } d_1 = \frac{\ln \frac{S_t}{K} + (r + \frac{\sigma^2}{2})t}{\sigma\sqrt{t}}$$

$$\text{and } d_2 = d_1 - \sigma\sqrt{t}$$

## Black-Scholes (BS) Model:

$C$  = call option price

$N$  = CDF of the normal distribution

$S_t$  = spot price of an asset

$K$  = strike price

$r$  = risk-free interest rate

$t$  = time to maturity

$\sigma$  = volatility of the asset

This model makes many assumptions (especially of volatility) and often mismatches empirical findings in different options.  
-- Especially not practical for American options



## How?

We used LSTM(model1) and MLP(model2,model3) architectures to predict the options prices and got results that outperform the Black-Scholes model significantly.

1. Data
2. Models
3. TFX
4. Results



# Dataset

- Stock data from yahoo finance

<https://finance.yahoo.com/>

- Option data from yahoo finance option

<https://finance.yahoo.com/quote/WIMI/options?p=WIMI>

- Treasury Rate

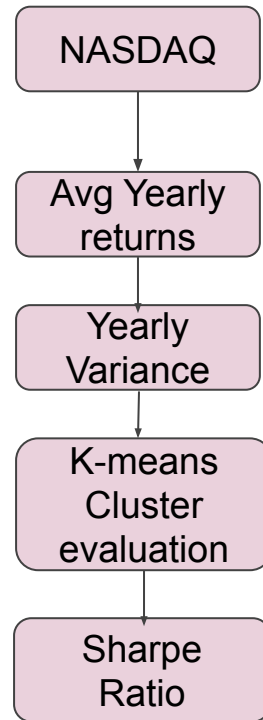
<https://home.treasury.gov/>

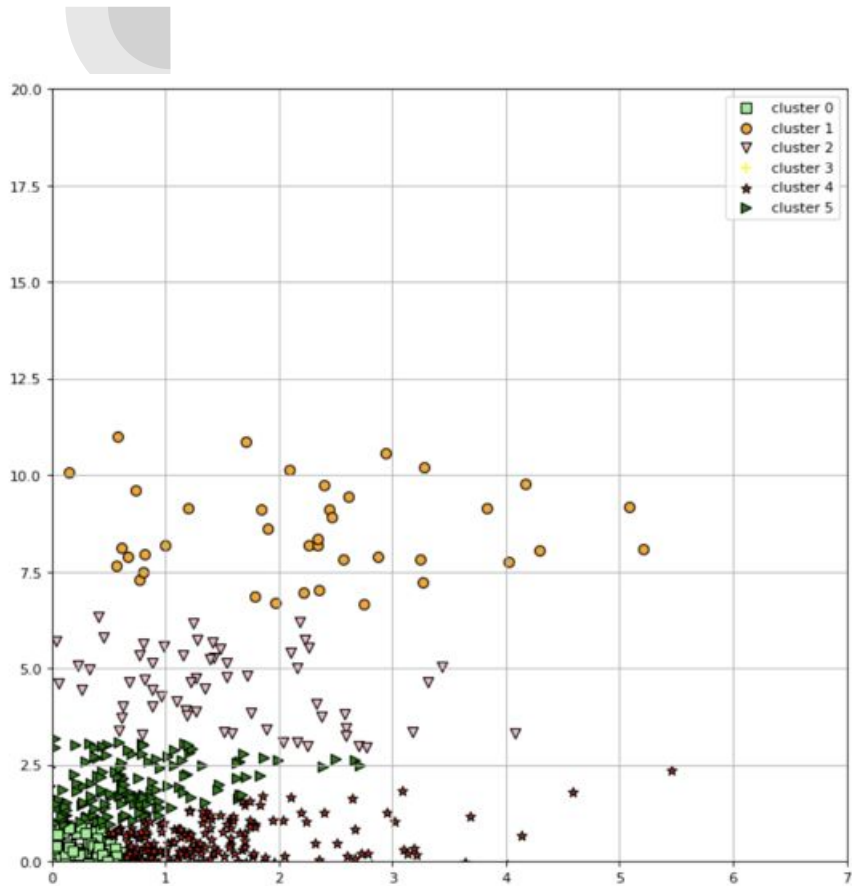


# Stock Picking

How to choose the best stocks?

## Strategy





# Applying K-means Clustering

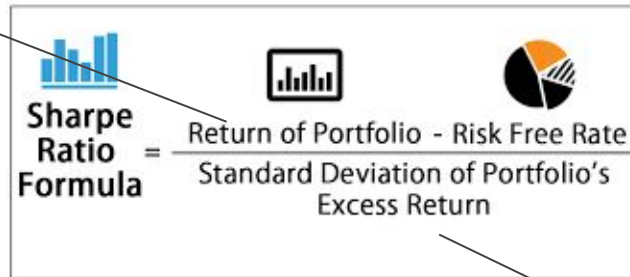
Cluster '0' further  
reduced down to 6  
clusters-

	cluster	avg_yearly_returns	yearly_variance	Name
0	3	-0.380375	0.096733	208
1	0	-0.007581	0.063594	984
2	5	0.105799	0.408778	204
3	2	0.284402	0.842208	75
4	4	0.359156	0.130867	246
5	1	0.560519	1.522883	40



# Sharpe Ratio for evaluating the best Stocks Cluster

Avg yearly returns



The diagram illustrates the Sharpe Ratio Formula. It features three icons at the top: a blue bar chart, a black bar chart, and a pie chart. The formula is presented as follows:

$$\text{Sharpe Ratio Formula} = \frac{\text{Return of Portfolio} - \text{Risk Free Rate}}{\text{Standard Deviation of Portfolio's Excess Return}}$$

Arrows point from the text labels to the corresponding parts of the formula: 'Avg yearly returns' points to 'Return of Portfolio', '3-years daily U.S. yield rates' points to 'Risk Free Rate', and 'Yearly variance' points to 'Standard Deviation of Portfolio's Excess Return'.

3-years daily U.S.  
yield rates  
(<https://home.treasury.gov/>)

Yearly variance

	cluster	avg_yearly_returns	yearly_variance	sharpe_ratio	Name
0	0	0.289101	0.067665	1.677841	137
1	1	0.326611	0.224425	0.649311	80
2	2	0.779880	0.171351	2.216106	29



# Highest Sharpe Ratio

	Name	avg_yearly_returns	yearly_variance	cluster	avg_risk_free_rate	std_dev	sharpe_ratio
1105	NSTG	0.841255	0.022083	2	0.021367	0.148602	5.517324
1646	VCYT	1.015007	0.057058	2	0.021367	0.238869	4.159773
890	JYNT	0.728127	0.037189	2	0.021367	0.192844	3.664930
640	FRPT	0.921197	0.069974	2	0.021367	0.264525	3.401683
1335	RGEN	0.714409	0.058437	2	0.021367	0.241738	2.866914
1474	SPSC	0.604935	0.046052	2	0.021367	0.214596	2.719375
968	LULU	0.674161	0.057792	2	0.021367	0.240400	2.715447
606	FIVN	0.815347	0.103550	2	0.021367	0.321792	2.467373
81	AMD	1.044455	0.189414	2	0.021367	0.435217	2.350752
484	DXCM	0.794039	0.108452	2	0.021367	0.329321	2.346259



# How to get options data for free?

```
: from bs4 import BeautifulSoup
import requests
import pandas as pd
from datetime import datetime
import time
```

```
: def main():
    print("Hello World!")
    if __name__ == '__main__':
        main()
```

```
: data_url = "https://finance.yahoo.com/quote/SPY/options"
data_html = requests.get(data_url).content
print(data_html)
```

<https://www.freecodecamp.org/news/how-i-get-options-data-for-free-fba22d395cc8/>

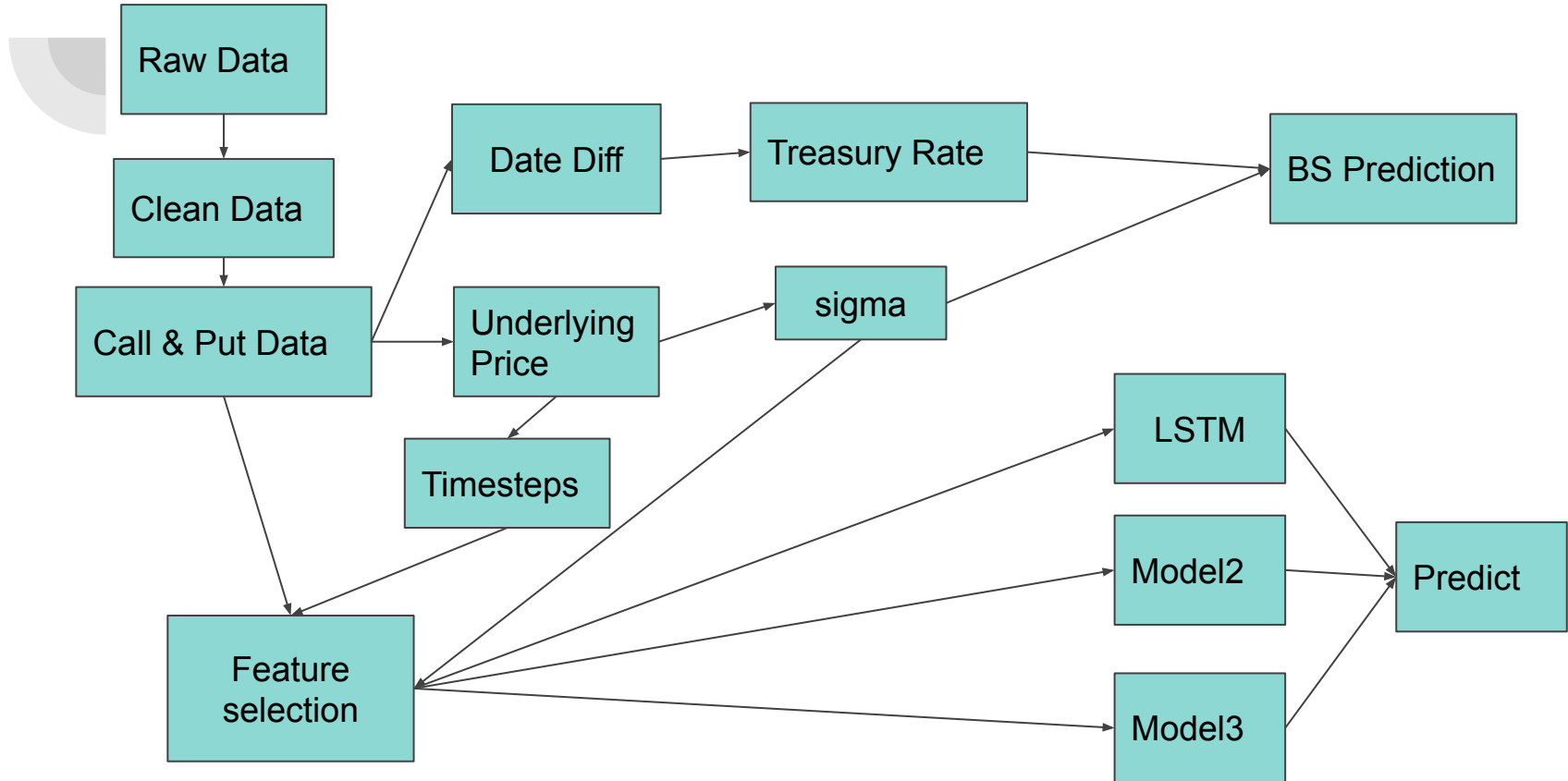
# Definitions





































df.head()

UnderlyingSymbol	UnderlyingPrice	Exchange	OptionSymbol	Blank	Type	Expiration	DataDate	Strike	Last	Bid	Ask	Volume	OpenInterest
ADSK	107.12	W	ADSK180105C00060000	NaN	call	01/05/2018	1/2/2018 04:00:00 PM	60.0	0.0	44.85	49.50	0.0	0.0

- \* Underlying: The stock, index, or ETF symbol
- \* Underlying\_price: The last traded price at the time of the option quote.
- \* Option type: Call or put Expiration The expiration date of the option.
- \* Expiration date: The date of the expiration
- \* Strike: The strike of the option
- \* Bid: The bid price of the option
- \* Ask: The ask price of the option
- \* Volume: The number of contracts traded
- \* Open interest : always a day behind. The OCC changes this number at 3:00AM every morning and the number does not change through the day

# Data Preprocessing



 call_X_test.csv	 call_df_21.csv	 dataset_2.csv	 dataset_1.csv	 call_df_21.tfrecords	 model3_call21_WIX_200.h5
  model2_call21_WIX.h5	  WIX_call-options-model2.csv	  WIX_put-options-black-sch...	  WIX_call-options-black-sch...	  WIX_option_treasury.csv	  JYNT_option_sigma.csv
  BLFS_option_sigma.csv	  WIX_option_sigma.csv	  VICR_option_sigma.csv	  VCYT_option_sigma.csv	  STAA_option_sigma.csv	  SPSC_option_sigma.csv
					

model\_lstm.png

model1\_lstm\_call\_1.h5

model1\_lstm\_call\_2.h5

model1\_lstm\_put\_1.h5

model2\_call\_sigma5\_all\_4...

model2\_call5\_all\_best.h5

model2\_call10\_all\_best.h5

model2\_call21\_all\_400.h5

model2\_call21\_all\_600.h5

model2\_call21\_all\_1200.h5

model2\_call21\_all\_2400.h5

model2\_call21\_all\_4000.h5

model2\_call21\_all\_best.h5

model2\_call21\_WIX.h5

model2\_put\_all\_300.h5

model2\_put\_all\_1500.h5

model2\_put\_all\_2400.h5

model2\_put21\_all\_best.h5

model2\_put21\_all5\_best.h5

model3\_call\_all\_300.h5

model3\_call\_all\_800.h5

model3\_call\_all\_1024\_400...

model3\_call\_all\_1024\_600...

model3\_call\_all\_2000.h5

model3\_call\_sigma5\_all\_1...

model3\_call5\_all\_best.h5

model3\_call21\_all\_best.h5

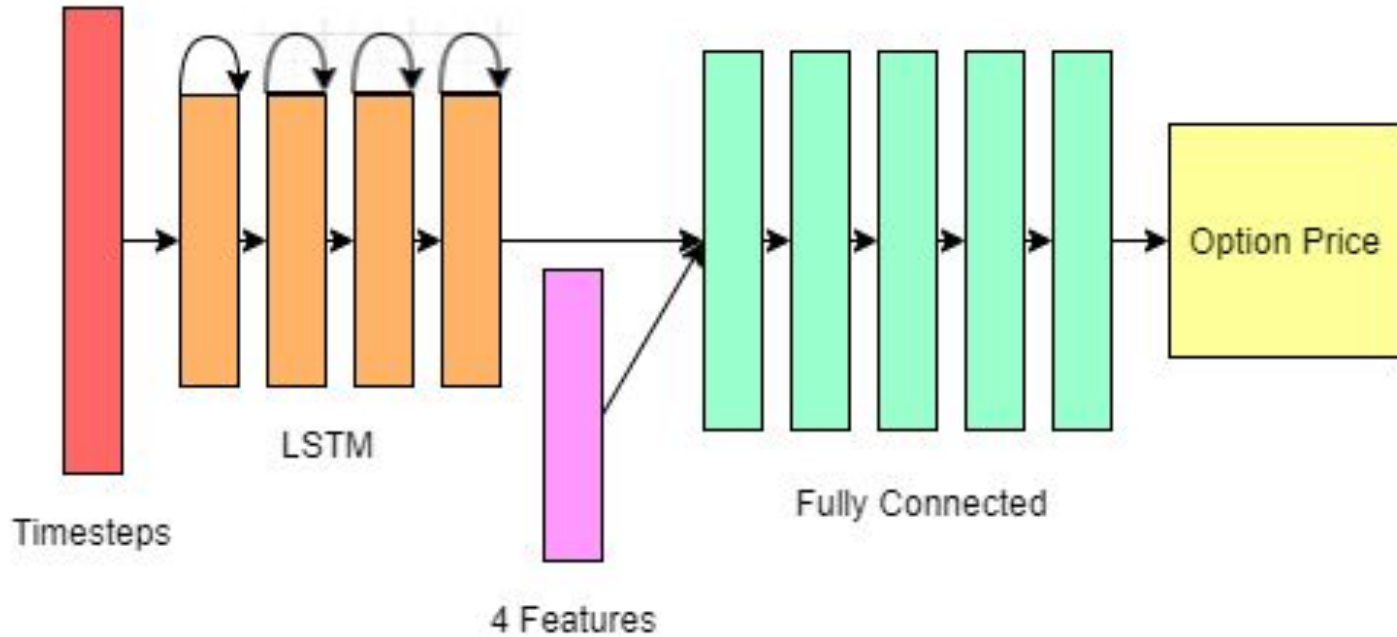
Input: [1?, 5]

# LSTM Features Timestep 5

	UnderlyingPrice	Strike	Bid	Ask	date_diff	treasury_rate	1	2	3	4	5
2344	11.91	0.5	11.35	11.45	3	1.33	11.91	12.02	12.1404	11.96	11.82
2345	11.91	1.0	10.90	10.95	3	1.33	11.91	12.02	12.1404	11.96	11.82
2346	11.91	1.5	10.35	10.45	3	1.33	11.91	12.02	12.1404	11.96	11.82
2347	11.91	2.0	9.90	9.95	3	1.33	11.91	12.02	12.1404	11.96	11.82
2348	11.91	2.5	9.35	9.45	3	1.33	11.91	12.02	12.1404	11.96	11.82
...	...	...	...	...	...	...	...	...	...	...	...
702371	148.52	185.0	16.40	19.20	534	2.00	30.45	33.87	33.4800	34.02	33.67
702372	148.52	190.0	15.00	17.80	534	2.00	30.45	33.87	33.4800	34.02	33.67
702373	148.52	195.0	13.90	16.40	534	2.00	30.45	33.87	33.4800	34.02	33.67
702374	148.52	200.0	12.90	15.20	534	2.00	30.45	33.87	33.4800	34.02	33.67
702375	148.52	210.0	10.10	13.20	534	2.00	30.45	33.87	33.4800	34.02	33.67



# LSTM

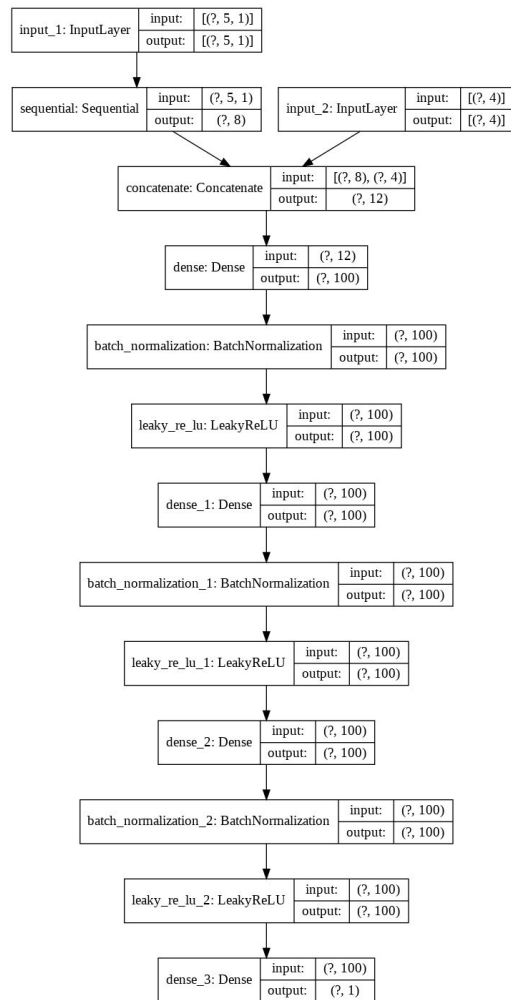


# Models- LSTM

```
call_model.summary()
```

Model: "functional\_1"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 5, 1)]	0	
sequential (Sequential)	(None, 8)	1952	input_1[0][0]
input_2 (InputLayer)	[(None, 4)]	0	
concatenate (Concatenate)	(None, 12)	0	sequential[0][0] input_2[0][0]
dense (Dense)	(None, 100)	1300	concatenate[0][0]
batch_normalization (BatchNorma	(None, 100)	400	dense[0][0]
leaky_re_lu (LeakyReLU)	(None, 100)	0	batch_normalization[0][0]
dense_1 (Dense)	(None, 100)	10100	leaky_re_lu[0][0]
batch_normalization_1 (BatchNor	(None, 100)	400	dense_1[0][0]
leaky_re_lu_1 (LeakyReLU)	(None, 100)	0	batch_normalization_1[0][0]
dense_2 (Dense)	(None, 100)	10100	leaky_re_lu_1[0][0]
batch_normalization_2 (BatchNor	(None, 100)	400	dense_2[0][0]
leaky_re_lu_2 (LeakyReLU)	(None, 100)	0	batch_normalization_2[0][0]
dense_3 (Dense)	(None, 1)	101	leaky_re_lu_2[0][0]
Total params: 24,753			





## Model2/Model3 Features Sigma 5

```
call_df_5.head()
```

	UnderlyingPrice	Strike	Bid	Ask	sigma_5	date_diff	treasury_rate
1042	12.28	5.5	6.75	6.85	0.033379	4	1.3
1043	12.28	6.0	6.20	6.35	0.033379	4	1.3
1044	12.28	6.5	5.75	5.85	0.033379	4	1.3
1045	12.28	7.0	5.25	5.35	0.033379	4	1.3
1046	12.28	7.5	4.70	4.85	0.033379	4	1.3

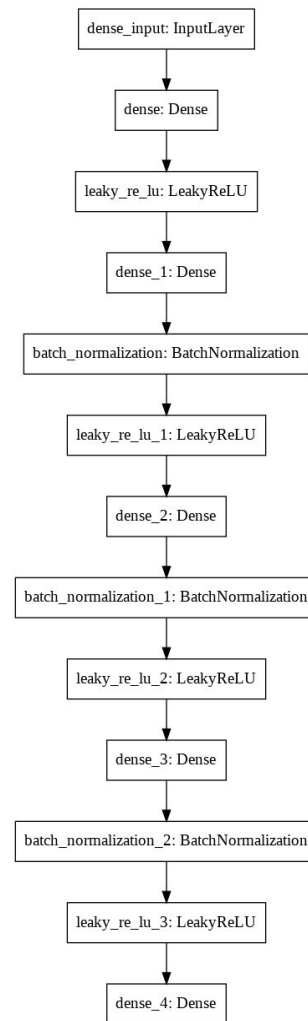
```
for day in [5, 10, 21, 30]:  
    df_underlying['sigma_'+str(day)] = df_underlying['UnderlyingPrice'].rolling(day).apply(lambda x: (np.diff(x) / x[:-1]).std())
```

# Models- Model2/3

Same architecture as Model2 ,but Model3 predict both ask and bid prices (Two output can be used for option market maker).

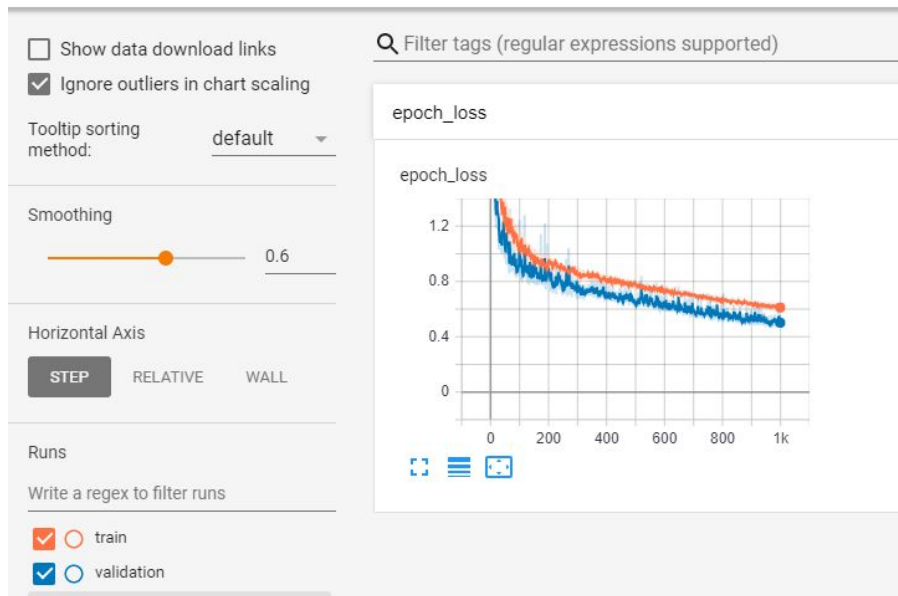
Model: "sequential\_5"

Layer (type)	Output Shape	Param #
dense_13 (Dense)	(None, 400)	2400
leaky_re_lu_8 (LeakyReLU)	(None, 400)	0
dense_14 (Dense)	(None, 400)	160400
batch_normalization (BatchNo	(None, 400)	1600
leaky_re_lu_9 (LeakyReLU)	(None, 400)	0
dense_15 (Dense)	(None, 400)	160400
batch_normalization_1 (Batch	(None, 400)	1600
leaky_re_lu_10 (LeakyReLU)	(None, 400)	0
dense_16 (Dense)	(None, 400)	160400
batch_normalization_2 (Batch	(None, 400)	1600
leaky_re_lu_11 (LeakyReLU)	(None, 400)	0
dense_17 (Dense)	(None, 2)	802
Total params: 489,202		
Trainable params: 486,802		
Non-trainable params: 2,400		



## LSTM2\_all\_put10

n\_units = 400, layers = 4, n\_batch = 1024, n\_epochs = 1000, learning\_rate = 1e-4



## LSTM

Put5

<https://tensorboard.dev/experiment/IN9XplUvRbqhaVBvCPIISg/>

Put10

<https://tensorboard.dev/experiment/oOljEuYVSvOLPDi17AtRww/#scalars>

Put21

<https://tensorboard.dev/experiment/h5KIjLjiQPmdk9cvD9JD6A/>

Call10

<https://tensorboard.dev/experiment/9tG0Hv68TmyaI7NBnbr2Ow/>

Call21

<https://tensorboard.dev/experiment/ZE4cErmZRXXKvUJRpHbjHO/#scalars>

### Model2\_all\_put

n\_units = 400, layers = 4, n\_batch = 1024, n\_epochs = 2400, learning\_rate = 1e-5

- ☐ Show data download links
- ☒ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing

0.946

Horizontal Axis

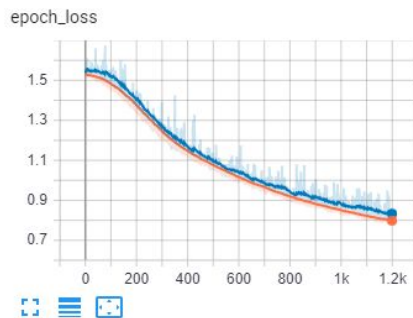
STEP RELATIVE WALL

Runs

Write a regex to filter runs

Filter tags (regular expressions supported)

epoch\_loss



## Model2:

Call 5

<https://tensorboard.dev/experiment/Bgx7f38NSgOtJQdNMXI9JA/#scalars>

call10

<https://tensorboard.dev/experiment/y9MuyU77Tymi0Uz1Yb0NSA/#scalars>

Call 21

<https://tensorboard.dev/experiment/y9MuyU77Tymi0Uz1Yb0NSA/>

put5

<https://tensorboard.dev/experiment/L208zOrlRziyfgiEm9RjjQ/#scalars>

Put21

<https://tensorboard.dev/experiment/L208zOrlRziyfgiEm9RjjQ/>

## Model3:

call5

<https://tensorboard.dev/experiment/X4dEGe3NQbOBOU5vFzHQuw/>

Call21

<https://tensorboard.dev/experiment/V2WF49BDQIK1spQ21eedfA/>

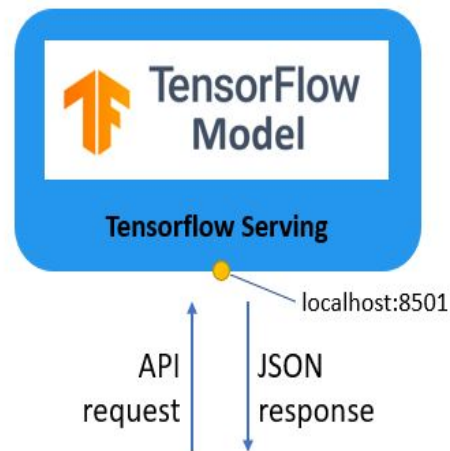
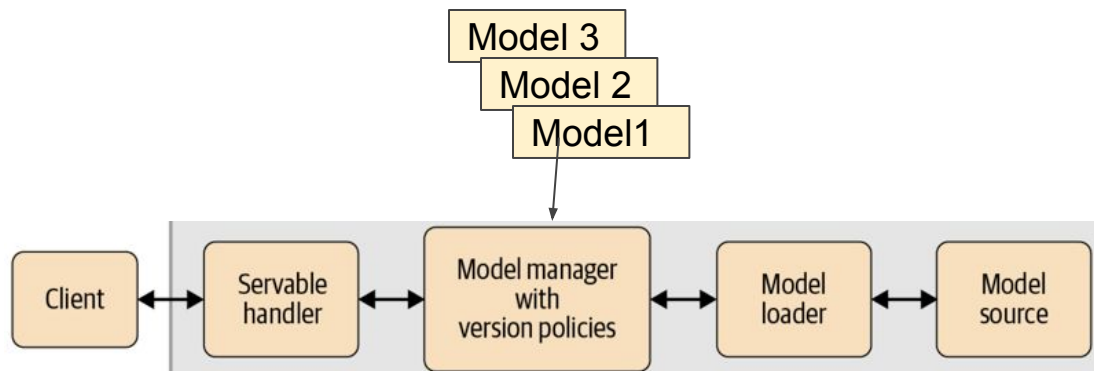
Put5

<https://tensorboard.dev/experiment/Ry5C4BGKRIKpndYhO4sdgg/>

Put21

<https://tensorboard.dev/experiment/9zzUE8NcQoySec6DarEo9g/>

# TFX Architecture



```
predictions = json.loads(json_response.text)['predictions']
```

# Convert Data to TFRecord

```
original_data_file = "/content/gdrive/MyDrive/DL_project/dataset/call_df_21.csv"
tfrecords_filename = "/content/gdrive/MyDrive/DL_project/dataset/call_df_21.tfrecords"
tf_record_writer = tf.io.TFRecordWriter(tfrecords_filename)

with open(original_data_file) as csv_file:
    reader = csv.DictReader(csv_file, delimiter=",", quotechar='"')
    for row in tqdm(reader):
        example = tf.train.Example(
            features=tf.train.Features(
                feature={
                    "UnderlyingPrice": _bytes_feature(row["UnderlyingPrice"]),
                    "Strike": _bytes_feature(row["Strike"]),
                    "Bid": _bytes_feature(row["Bid"]),
                    "Ask": _bytes_feature(row["Ask"]),
                    "sigma_21": _bytes_feature(row["sigma_21"]),
                    "date_diff": _bytes_feature(row["date_diff"]),
                    "treasury_rate": _bytes_feature(row["treasury_rate"]),
                }
            )
        )
        tf_record_writer.write(example.SerializeToString())
    tf_record_writer.close()
```

49833it [00:04, 12307.15it/s]



# Interactive Context

```
context = InteractiveContext()
examples = external_input(_data_root)
example_gen =
CsvExampleGen(input=examples)
context.run(example_gen)
```

INFO:absl:MetadataStore with DB connection initialized

▼ ExecutionResult at 0x7efe1dc012b0

.execution\_id 1

.component

▼ CsvExampleGen at 0x7efe1ffc9b0

.inputs {}

.outputs ['examples'] ► Channel of type 'Examples' (1 artifact) at 0x7efe1dc14588

.exec\_properties

['input_base']	/tmp/tfx-data4gidhajq
['input_config']	{ "splits": [ { "name": "single_split", "pattern": "*" } ] }
['output_config']	{ "split_config": { "splits": [ { "hash_buckets": 2, "name": "train" }, { "hash_buckets": 1, "name": "eval" } ] } }
['output_data_format']	6
['custom_config']	None
['span']	0
['version']	None
['input_fingerprint']	split:single_split,num_files:1,total_bytes:2884897,xor_checksum:1607385949,sum_checksum:1607385949
['_beam_pipeline_args']	[]

.component.inputs {}

.component.outputs

['examples'] ▼ Channel of type 'Examples' (1 artifact) at 0x7efe1dc14588

.type\_name Examples

.\_artifacts

[0] ▼ Artifact of type 'Examples' (uri: /tmp/tfx-interactive-2020-12-08T00\_05\_49.125751-nlyspvb7/CsvExampleGen/examples/1) at 0x7efe1dc145c0

.type <class 'tfx.types.standard\_artifacts.Examples'>

.uri /tmp/tfx-interactive-2020-12-08T00\_05\_49.125751-nlyspvb7/CsvExampleGen/examples/1

.span 0

.split\_names ["train", "eval"]

.version 0

# Interactive Context

## Artifact

```
feature {
  key: "Ask"
  value {
    float_list {
      value: 22.0
    }
  }
}
feature {
  key: "Bid"
  value {
    float_list {
      value: 20.399999618530273
    }
  }
}
feature {
  key: "Strike"
  value {
    float_list {
      value: 40.0
    }
  }
}
feature {
  key: "UnderlyingPrice"
  value {
    float_list {
      value: 61.04999923706055
    }
  }
}
```

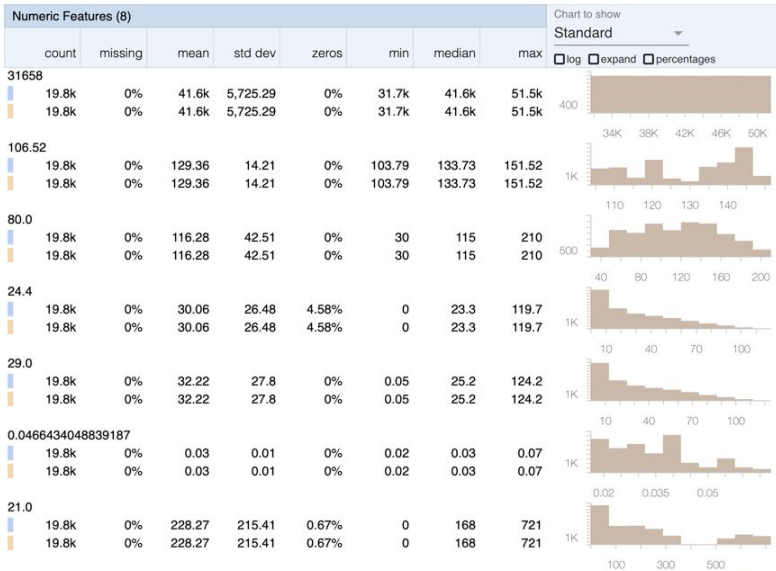
## Data SchemaGen

```
schema = tfdv.infer_schema(stats)
tfdv.display_schema(schema)
```

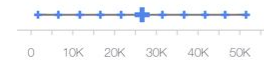
Feature name	Type	Presence	Valency	Domain
'Ask'	BYTES	required		-
'Bid'	BYTES	required		-
'Strike'	STRING	required		'Strike'
'UnderlyingPrice'	BYTES	required		-
'date_diff'	BYTES	required		-
'sigma_21'	BYTES	required		-
'treasury_rate'	BYTES	required		-

# TFX Data Visualization (TFDV)

☒ VAL\_DATASET ☒ TRAIN\_DATASET



count	missing	mean	std dev	zeros	min	median	max
374	0%	26.03	23.75	0%	0.05	19.4	121.5



Ask

16.6k	0%	26.03	23.75	0%	0.05	19.4	121.5
-------	----	-------	-------	----	------	------	-------

Bid

16.6k	0%	23.88	22.57	6.17%	0	17.4	117.3
-------	----	-------	-------	-------	---	------	-------

Strike

16.6k	0%	102.09	40.32	0%	30	100	210
-------	----	--------	-------	----	----	-----	-----

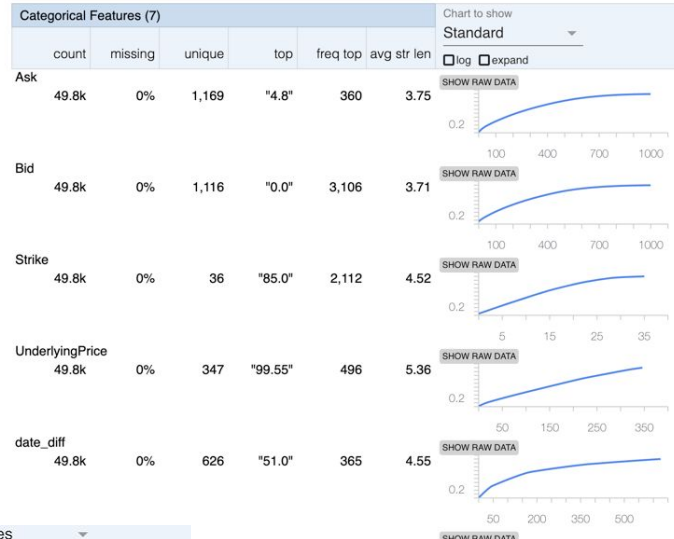
Sort by

Feature order

☐ Reverse order

Feature search (regex enabled)

Features: ☒ string(7)



count	missing	mean	std dev	zeros	min	median	max
374	0%	26.03	23.75	0%	0.05	19.4	121.5



Ask

16.6k	0%	26.03	23.75	0%	0.05	19.4	121.5
-------	----	-------	-------	----	------	------	-------

Bid

16.6k	0%	23.88	22.57	6.17%	0	17.4	117.3
-------	----	-------	-------	-------	---	------	-------

Strike

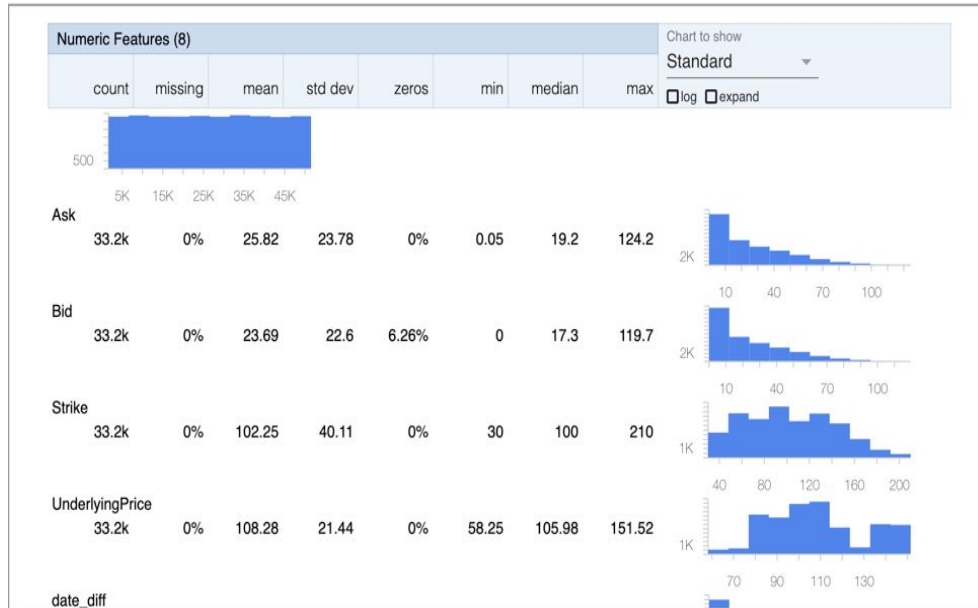
16.6k	0%	102.09	40.32	0%	30	100	210
-------	----	--------	-------	----	----	-----	-----

# Data StatisticsGen



## Anomalies in Data

```
context.show(statistics_gen.outputs['statistics'])
```



Feature name	Anomaly short description	Anomaly long description
'treasury_rate'	Column dropped	Column is completely missing
'\29.0\"'	New column	New column (column in data but not in schema)
'\80.0\"'	New column	New column (column in data but not in schema)
'\24.4\"'	New column	New column (column in data but not in schema)
'date_diff'	Column dropped	Column is completely missing
'31658'	New column	New column (column in data but not in schema)
'\106.52\"'	New column	New column (column in data but not in schema)
'Bid'	Column dropped	Column is completely missing
'0.0466434048839187\"'	New column	New column (column in data but not in schema)
'Ask'	Column dropped	Column is completely missing
'UnderlyingPrice'	Column dropped	Column is completely missing
'\21.0\"'	New column	New column (column in data but not in schema)
'sigma_21'	Column dropped	Column is completely missing
'Strike'	Column dropped	Column is completely missing
'\2.36\"'	New column	New column (column in data but not in schema)



# TFX Model Serving

```
2020-12-07 00:54:58.210420: I external/org_tensorflow/tensorflow/cc/saved_model/loader.cc:199] Restoring SavedModel bundle.
2020-12-07 00:54:58.254873: I external/org_tensorflow/tensorflow/cc/saved_model/loader.cc:183] Running initialization op on SavedModel bundle a
2020-12-07 00:54:58.262050: I external/org_tensorflow/tensorflow/cc/saved_model/loader.cc:303] SavedModel load for tags { serve }; Status: succ
2020-12-07 00:54:58.263058: I tensorflow_serving/servables/tensorflow/saved_model_warmup_util.cc:59] No warmup data file found at /tmp/1/assets
2020-12-07 00:54:58.263216: I tensorflow_serving/core/loader_harness.cc:87] Successfully loaded servable version {name: first_model version: 1}
2020-12-07 00:54:58.264399: I tensorflow_serving/model_servers/server.cc:367] Running gRPC ModelServer at 0.0.0.0:8500 ...
[warn] getaddrinfo: address family for nodename not supported
2020-12-07 00:54:58.265214: I tensorflow_serving/model_servers/server.cc:387] Exporting HTTP/REST API at:localhost:8501 ...
[evhttp_server.cc : 238] NET LOG: Entering the event loop ...
```

## Predictions

```
'{"signature_name": "serving_default", "instances": [[141.96, 190.0, 0.040351047419522064, 225.0, 2.22], [82.85, 70.0, 0.02629
471439129068, 37.0, 1.64], [97.4, 155.0, 0.03228772966546013, 571.0, 2.54], [110.11, 135.0, 0.06315234590095266, 42.0, 2.41],
[109.1, 95.0, 0.016403341833589692, 140.0, 2.52], [137.46, 115.0, 0.026066182636784167, 4.0, 2.41], [79.65, 60.0, 0.0264284369
79249875, 228.0, 1.86], [105.2, 130.0, 0.02488153995771937, 178.0, 2.19], [79.35, 75.0, 0.02112959870762864, 318.0, 2.06], [9
7.4, 45.0, 0.03228772966546013, 116.0, 1.93], [111.96, 70.0, 0.04309717861046268, 17.0, 2.14], [106.3, 60.0, 0.036272658671604
59, 15.0, 1.98], [96.6, 135.0, 0.06593577053311157, 63.0, 2.36], [90.65, 145.0, 0.05369590696567412, 744.0, 2.5], [108.15, 95.
0, 0.01953874168830817, 130.0, 2.46], [101.34, 120.0, 0.05211642321528565, 93.0, 2.45], [89.4, 40.0, 0.02907330589461383, 93.
0, 1.81], [141.96, 55.0, 0.040351047419522064, 225.0, 2.22], [125.75, 65.0, 0.0567472996325723, 696.0, 2.5], [122.65, 170.0,
0.04381723541187575, 191.0, 2.46], [112.0, 75.0, 0.044291377964058416, 480.0, 2.6], [95.0, 60.0, 0.039871988179493965, 535.0,
2.44], [142.03, 190.0, 0.02304807360549109, 256.0, 2.46], [119.3, 60.0, 0.041100395508881706, 108.0, 2.42], [107.6, 145.0, 0.0
2658037020883468, 576.0, 2.56], [82.05, 40.0, 0.03607495935958746, 248.0, 2.09], [106.3, 75.0, 0.03627265867160459, 15.0, 1.9
8], [124.48, 95.0, 0.035416602172033324, 276.0, 2.45], [83.425, 30.0, 0.03688202209474797, 254.0, 2.05], [146.01, 110.0, 0.040
34332393473633, 11.0, 2.3], [110.85, 90.0, 0.02266813029334776, 86.0, 2.01], [83.18, 70.0, 0.05119926052739802, 4.0, 2.36], [1
08.89, 85.0, 0.033616104014165364, 59.0, 2.46], [139.47, 100.0, 0.02178936629123049, 549.0, 1.87], [77.4, 35.0, 0.022976734322
1936, 291.0, 2.08], [144.62, 155.0, 0.02990831354276945, 108.0, 2.2], [112.45, 65.0, 0.046253993306802715, 485.0, 2.58], [148.
12, 105.0, 0.03955871329297976, 28.0, 2.16], [140.94, 80.0, 0.036935548741476684, 590.0, 1.83], [86.45, 110.0, 0.0358836905231
-----
```

# MSE Results

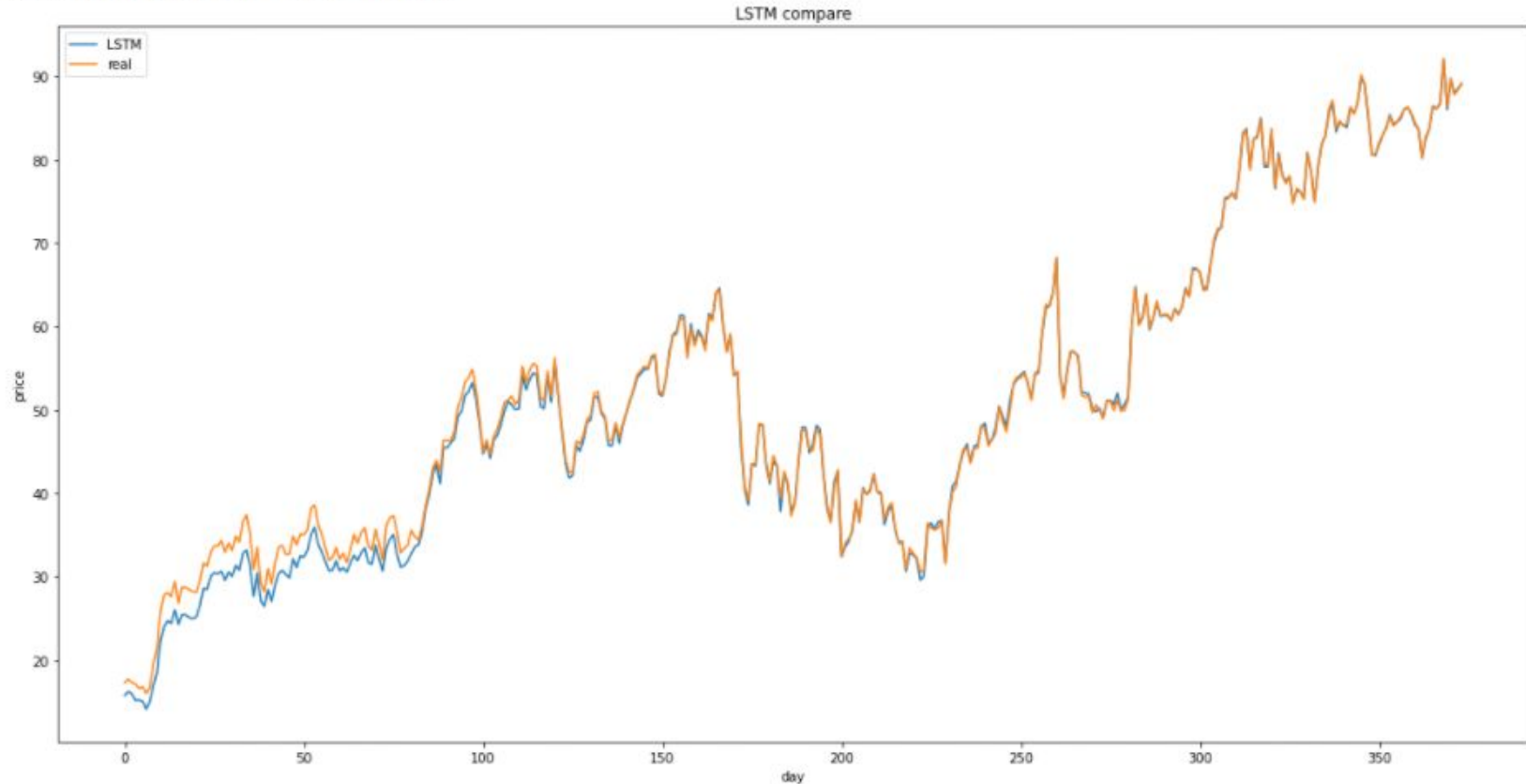
model	learning rate	Batch Size	epoch	sigma	MSE
LSTM call	0.0001	1024	1000	21	0.58
LSTM put	0.0001	1024	1000	10	0.6089
Model2 call	1.00E-05	1024	4000	21	0.8284
Model2 put	1.00E-05	1024	3000	5	0.821
Model3 call	1.00E-05	1024	2000	5	0.4618
Model3 put	1.00E-05	1024	4000	21	0.8204

model	sigma5	sigma10	sigma21
LSTM call	0.8	0.6206	0.58
LSTM put	0.69	0.6089	0.6206
Model2 call	0.833	0.8442	0.8284
Model2 put	0.821	0.9118	0.832
Model3 call	0.4618	0.8623	0.7406
Model3 put	0.8602	1.0232	0.8204
BS Model call	Nan	Nan	23.2528235
Bs Model put	Nan	Nan	23.7103045

model	MSE
LSTM call	0.58
LSTM put	0.6089
Model2 call	0.8284
Model2 put	0.832
Model3 call	0.4618
Model3 put	0.8204
BS Model call	23.2528235
Bs Model put	23.7103045

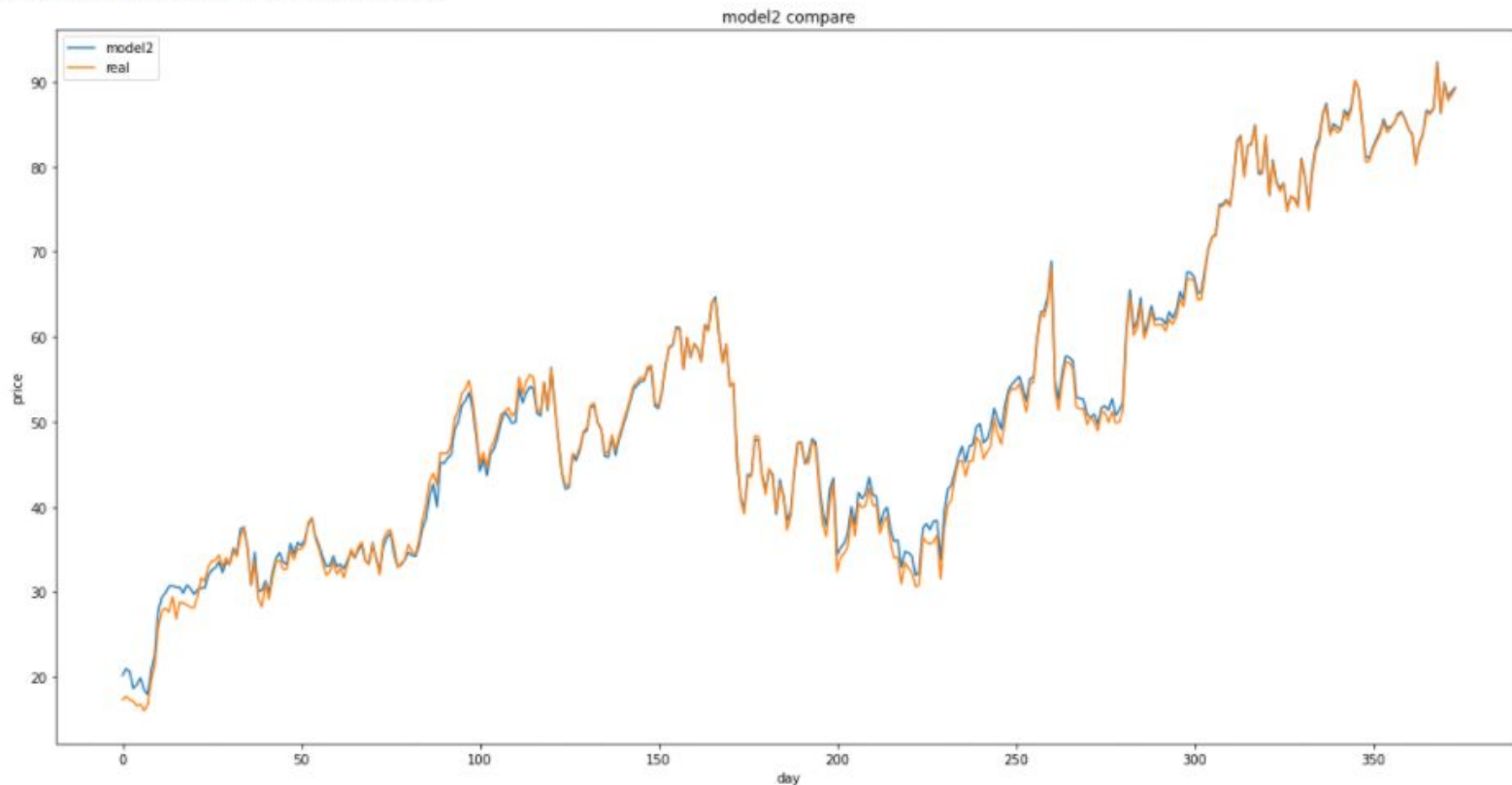
# LSTM Mean Price Prediction

<matplotlib.legend.Legend at 0x7f386188ef28>



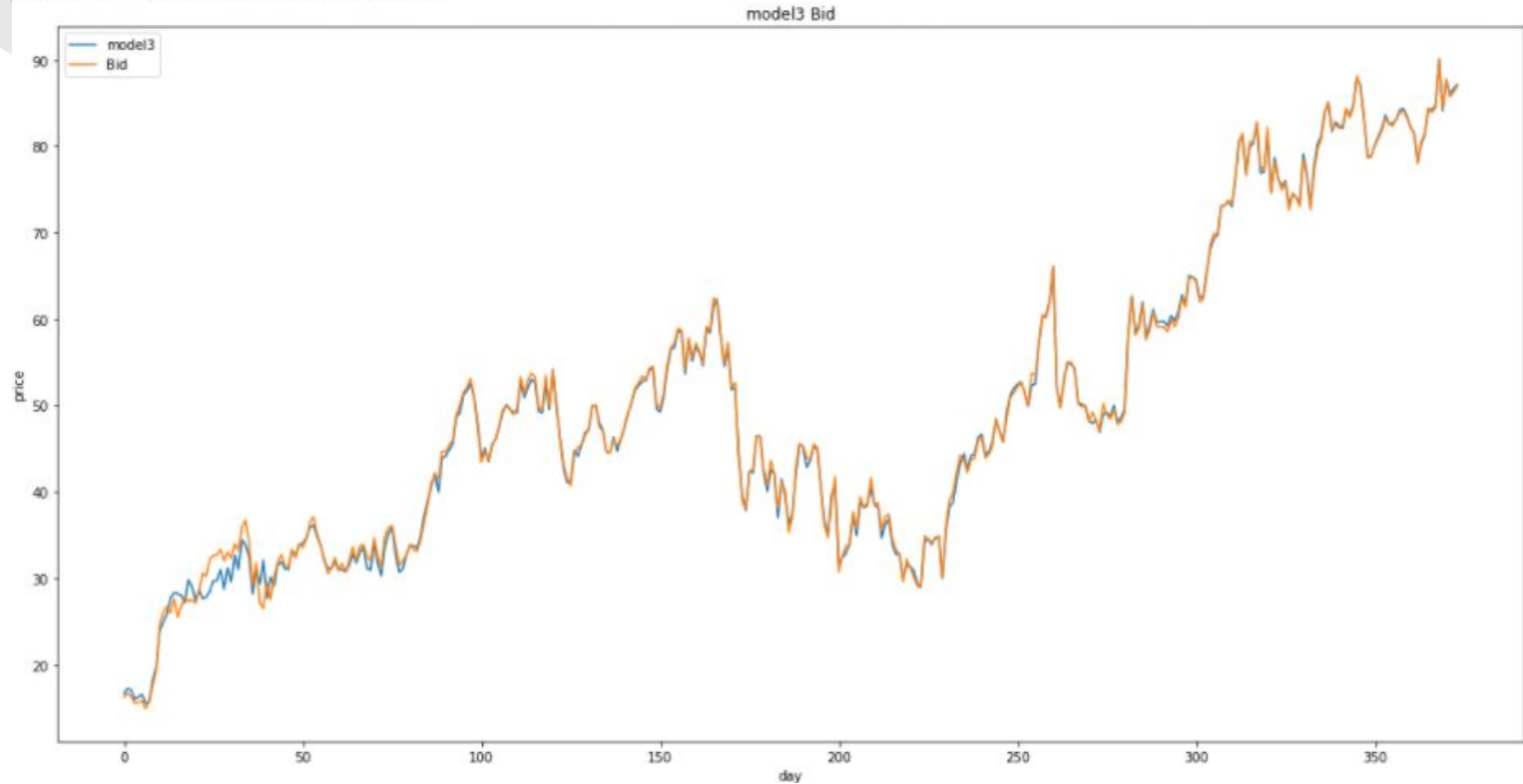
# Model2 Mean Price Prediction

<matplotlib.legend.Legend at 0x7f38626932b0>

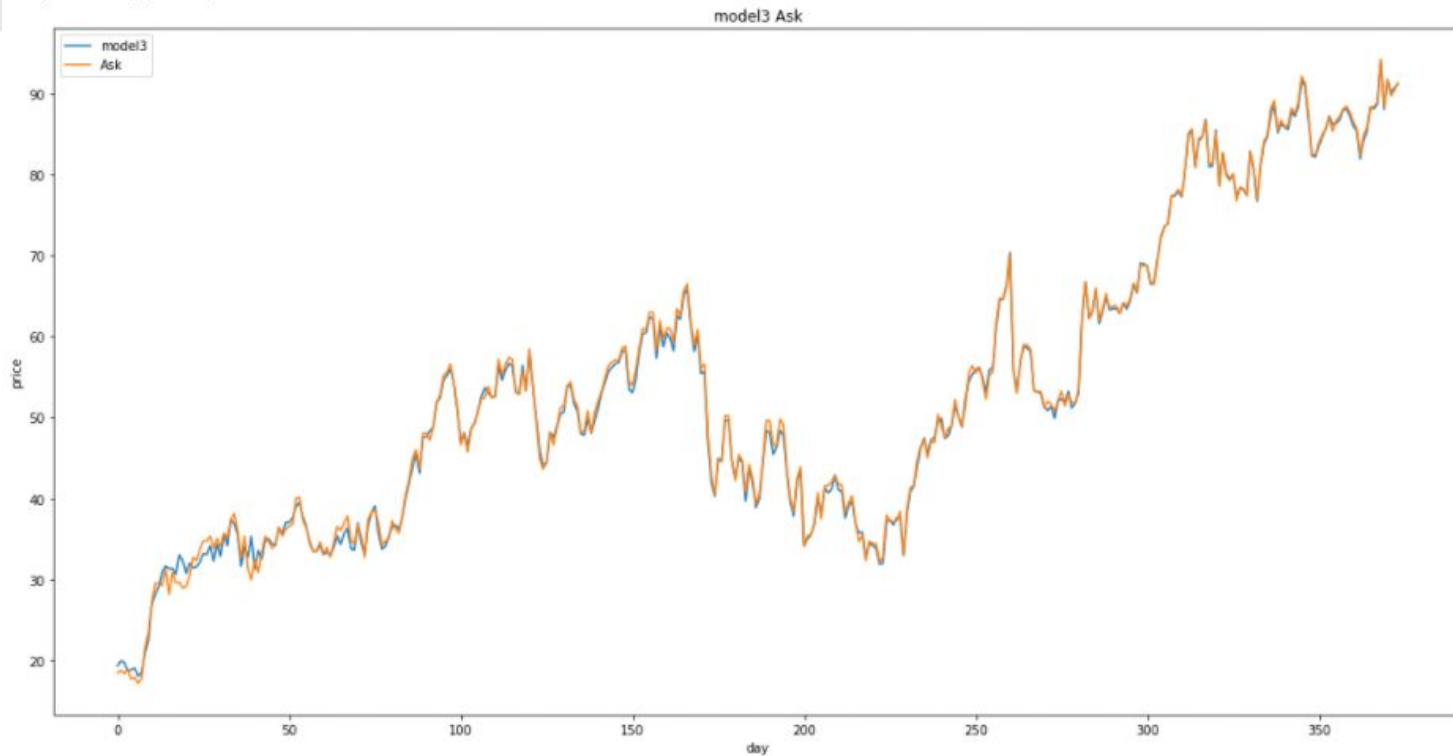




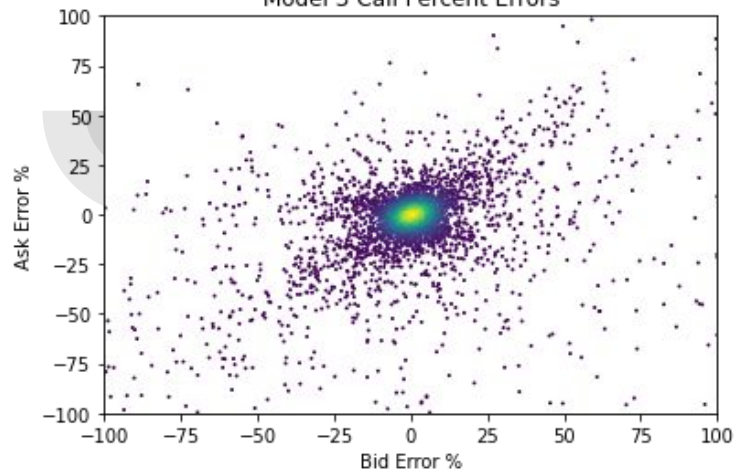
# Model 3 Bid price Prediction



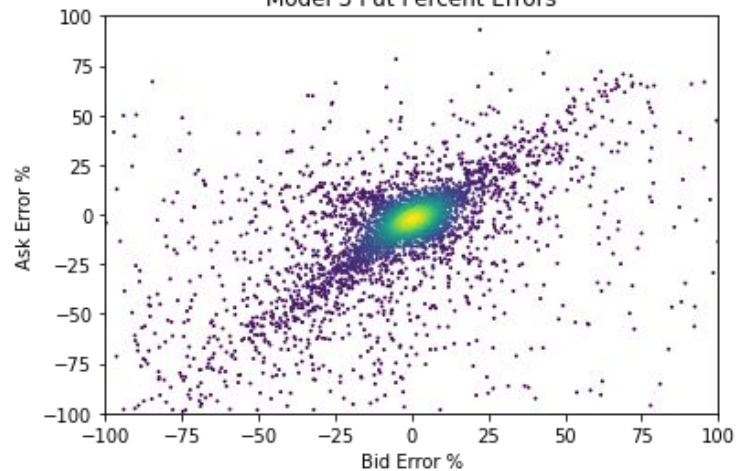
# Model 3 on Ask price Prediction



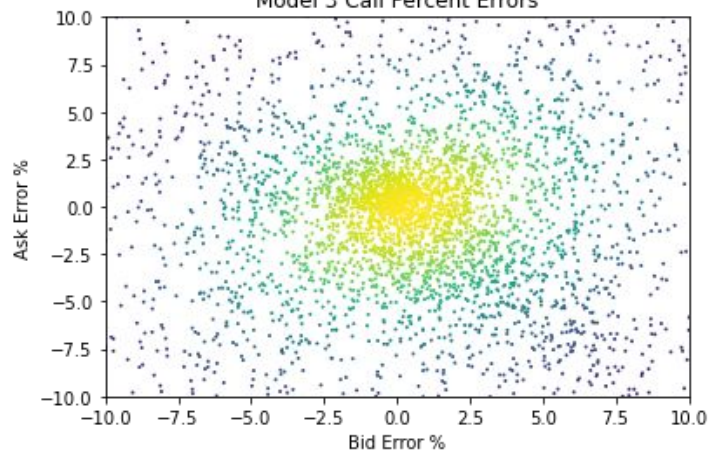
Model 3 Call Percent Errors



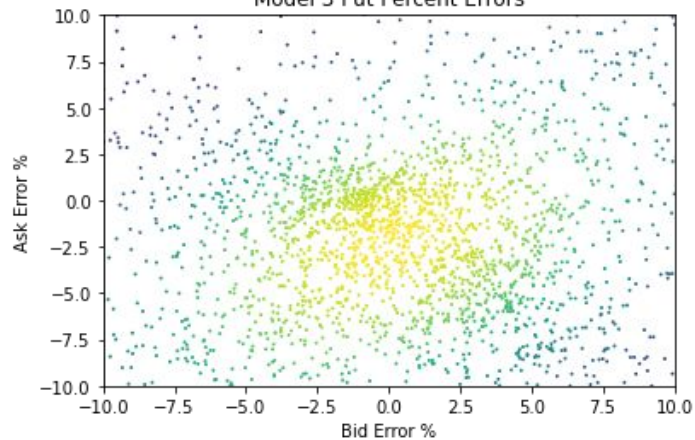
Model 3 Put Percent Errors



Model 3 Call Percent Errors



Model 3 Put Percent Errors





**Github:**

[https://github.com/zjzsu2000/CMPE297\\_AdvanceDL\\_Project](https://github.com/zjzsu2000/CMPE297_AdvanceDL_Project)



# Future Work

- **Stocks filter** — Using the stocks with better Sharpe Rate
- **Option filter**
  - Not using the options deep out of the money(The Deep-OTM options are more volatile)
  - Not using the options expired in 7 days (The options closer to the expiry date are more volatile)
- **More models, More data, More training, More metric**
- **Apply to the real trading**
- **Fine-tuning with more hyperparameters**