нетология

Прогноз цены биткойна на момент закрытия дневных торгов

Александр Сибагатов

1.ПОСТАНОВКА ЗАДАЧИ

Об исследовании

- **Объектами исследования:**
  - является Значения биткойнов в долларах США с 01.01.2017 года по 15.11.2019 года.

- **Предметом исследования:**
  - является набор данных "Bitcoin Historical USD Price".

- **Целью работы:**
  - является работа с временным рядом и построение модели нейронной сети для его прогноза.

Сферы применения и актуальность работы

- Финансовый сектор
- Показания датчиков
- Объемы продаж/производства
- Телеметрия it-систем
- и т.д.

В работе будут рассмотрены следующие задачи

- Исследование данных
- Построение временного ряд
- Построение графика Автокорреляции
- Создание набора данных для обучения
- Создание нейронной сети
- Разделение данных
- Масштабирование данных
- Обучение и проверка модели
- Прогнозирование на тестовом наборе данных
- Оценка качества модели на тестовой выборке
- Построение графика прогноза

Целевые метрики

MSE (mean squared error)

$$\mathrm{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2.$$

R^2 (coefficient of determination)

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

, где $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$ и $\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} \epsilon_i^2$,

2. АНАЛИЗ

О наборе данных

Значения биткойнов в долларах США с 1 января 2017 года по 15 ноября 2019 года, загруженные с сайта Yahoo Finance с однодневным разрешением.

По набору данных

1. столбец datetime ***Date***;
2. начальное значение торгового дня ***Open***;
3. конечное значение в любое время, которое трейдеры должны назвать днем ***Close***;
4. самое высокое ***Hight*** и самое низкое значения ***Low*** за день;
5. ***Adj Close*** скорректированная рыночная стоимость закрытия;
6. ***Volume*** полный объем.

Описательная статистика target значения ‘Close’

	Close
count	1049.000000
mean	6276.268179
std	3601.456029
min	777.760000
25%	3631.040000
50%	6377.780000
75%	8586.470000
max	19497.400000

- В наборе данных приблизительно 1050 дней наблюдений *(count = 1049)*.
- Максимальная стоимость биткойна составляло около 19 тыс. Долларов США.
- Минимальное падение курса в диапазоне семиста Долларов США.

Построение временного ряда

Построение графика автокорреляции

По предварительному анализу:

- target значение ‘Close’

- колонки Adj Close и Volume исключены

- по графику временного ряда, основной пик цены происходит в Июле

- по графику автокорреляции, наиболее коррелированные значения в диапазоне 15 lags

Дальнейшие шаги:

- Создать новый набор данных используя последние 15 значений в качестве входных данных

- Построить модель

- Разделить данные на train, val, test

- Масштабировать данные для уменьшения дисперсии

- Обучить и протестировать модель

3. МЕТОДИКА РЕШЕНИЯ

Создание набора данных для обучения

	Date	Open	High	Low	Close	Adj Close	Volume
0	2017-01-01	963.66	1003.08	958.70	998.33	998.33	147775008
1	2017-01-02	998.62	1031.39	996.70	1021.75	1021.75	222184992
2	2017-01-03	1021.60	1044.08	1021.60	1043.84	1043.84	185168000
3	2017-01-04	1044.40	1159.42	1044.40	1154.73	1154.73	344945984
4	2017-01-05	1156.73	1191.10	910.42	1013.38	1013.38	510199008

1

```
def create_regressor_attributes
```

3

```
df_new = create_regressor_attributes
```

5

datetime	Close	Close_(t-1)	Close_(t-2)
2017-01-16	831.53	821.80	818.41
2017-01-17	907.94	831.53	821.80
2017-01-18	886.62	907.94	831.53
2017-01-19	899.07	886.62	907.94
2017-01-20	895.03	899.07	886.62

*
*
*

Close_(t-15)
998.33
1021.75
1043.84
1154.73
1013.38

Создание нейронной сети

Что будем строить

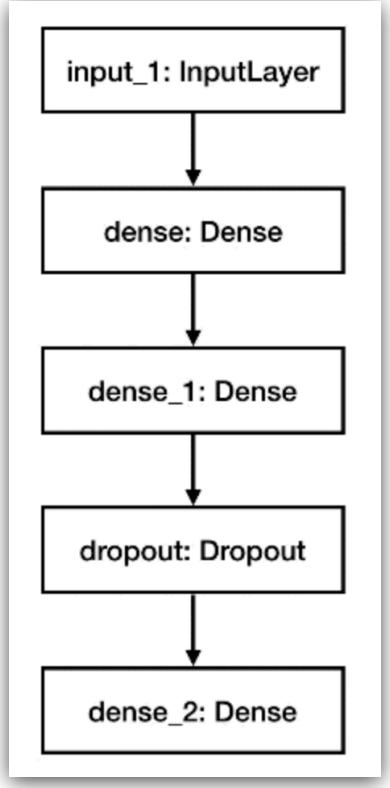
- Тренируем простой Многослойный Персептрон, который имеет входной слой с 15 узлами
- 2 скрытых слоя
- 60 узлов в каждом

Основные функции:

- Input
- Dense
- activation
- Dropout
- loss
- optimizer

Модель

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 15)]              0
_________________________________________________________________
dense (Dense)                (None, 60)                960
_________________________________________________________________
dense_1 (Dense)              (None, 60)                3660
_________________________________________________________________
dropout (Dropout)            (None, 60)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 61
=================================================================
Total params: 4,681
Trainable params: 4,681
Non-trainable params: 0
_________________________________________________________________
```



Разделение данных и Масштабирование данных

`X_train, y_train, X_valid, y_valid, X_test, y_test`

```
Shape of training inputs, training target: (932, 15) (932,)
Shape of validation inputs, validation target: (50, 15) (50,)
Shape of test inputs, test target: (52, 15) (52,)
```

`MinMaxScaler`

```
Target_scaler = MinMaxScaler(feature_range=(0.01, 0.99))
Feature_scaler = MinMaxScaler(feature_range=(0.01, 0.99))
```

Обучение и проверка модели

```
model.fit(
    x=X_train_scaled,
    y=y_train_scaled,
    batch_size=5,
    epochs=30,
    verbose=1,
    validation_data=(X_valid_scaled,
                     y_valid_scaled),
    shuffle=True)
```

```
Train on 932 samples, validate on 50 samples
Epoch 1/30
932/932 [==============================] - 1s 2ms/sample - loss: 0.0155 - val_loss: 0.0390
Epoch 2/30
932/932 [==============================] - 1s 601us/sample - loss: 0.0039 - val_loss: 0.0389
Epoch 3/30
932/932 [==============================] - 1s 639us/sample - loss: 0.0034 - val_loss: 0.0472
Epoch 4/30
932/932 [==============================] - 0s 531us/sample - loss: 0.0026 - val_loss: 0.0399
Epoch 5/30
932/932 [==============================] - 1s 584us/sample - loss: 0.0023 - val_loss: 0.0403
Epoch 6/30
932/932 [==============================] - 1s 578us/sample - loss: 0.0023 - val_loss: 0.0234
Epoch 7/30
932/932 [==============================] - 1s 583us/sample - loss: 0.0020 - val_loss: 0.0549
```

* * *

```
Epoch 23/30
932/932 [==============================] - 1s 588us/sample - loss: 0.0013 - val_loss: 0.0323
Epoch 24/30
932/932 [==============================] - 1s 565us/sample - loss: 9.3502e-04 - val_loss: 0.0344
Epoch 25/30
932/932 [==============================] - 1s 547us/sample - loss: 0.0011 - val_loss: 0.0409
Epoch 26/30
932/932 [==============================] - 1s 565us/sample - loss: 9.3423e-04 - val_loss: 0.0370
Epoch 27/30
932/932 [==============================] - 1s 561us/sample - loss: 0.0016 - val_loss: 0.0299
Epoch 28/30
932/932 [==============================] - 1s 575us/sample - loss: 9.4867e-04 - val_loss: 0.0331
Epoch 29/30
932/932 [==============================] - 1s 568us/sample - loss: 9.0551e-04 - val_loss: 0.0378
Epoch 30/30
932/932 [==============================] - 1s 589us/sample - loss: 0.0011 - val_loss: 0.0425
<tensorflow.python.keras.callbacks.History at 0x7f0955944d68>
```

Наши потери при проверке не сильно изменились, особенно по сравнению с потерями при обучении.

Мы можем приписать это небольшому количеству примеров обучения, которые у нас есть, в контексте модели, которую мы использовали.

4. РЕЗУЛЬТАТЫ

Прогнозирование на тестовом наборе данных

```
y_pred = model.predict(X_test_scaled)
```

Помним, что все наши входы и цели были уменьшены в диапазоне (0, 1). Таким образом, прогнозы также лежат в этом диапазоне. Нам нужно уменьшить их в обратном направлении.

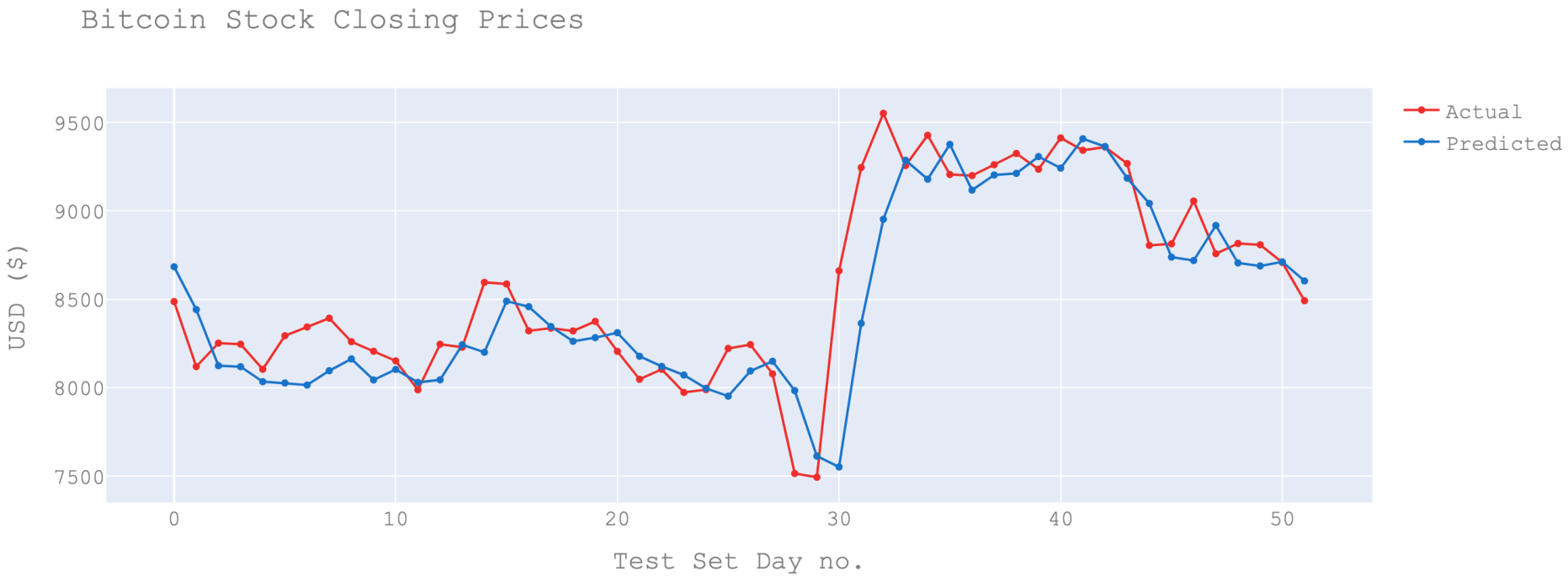
```
y_pred_rescaled = Target_scaler.inverse_transform(y_pred)
```

Оценка качества модели на тестовой выборке

```
y_test_rescaled =  Target_scaler.inverse_transform(y_test_scaled)
score = r2_score(y_test_rescaled, y_pred_rescaled)
print('R-squared score for the test set:', round(score,4))
```

```
R-squared score for the test set: 0.7222
```

Построение графика прогноза



5. ЗАКЛЮЧЕНИЕ

Выводы

- Задача прогноза временных рядов зависит от внутренних и внешних факторов во временном отрезке, что достаточно сильно приближает модели по временным рядам к категории моделей “черного ящика”.

- По прогнозу данного набора данных. В горизонте прогнозирования на один день, прогноз выглядит правдоподобно для задачи в которой присутствуют факторы времени и риска.

Куда дальше?

LSTM и CNN могут давать лучшие результаты, чем обычные NN на одних и тех же данных.

Литература и полезные ссылки:

- Анализ временных рядов с помощью Python
- Сравнение моделей временных рядов
- Holt-Winters Forecasting for Dummies (or Developers)
- Open Machine Learning Course. Topic 9. Part 1. Time series analysis in Python
- Open Machine Learning Course. Topic 9. Part 2. Predicting the future with Facebook Prophet
- Notes on regression and time series analysis
- The Time Series They Are a-Changing: Why all good models eventually fail
- Two key challenges for time series analysis
- Introduction to Interactive Time Series Visualizations with Plotly in Python
- Being Bayesian and thinking deep: time-series prediction with uncertainty
- Facing the ARIMA Model against Neural Networks
- Stationarity test for time series
- Time Series Decomposition and StatsModels Parameters
- Получение котировок акций при помощи Python
- TensorFlow Time Series Forecasting
- Descriptive statistics in Time Series Modelling
- Keras documentation
- Методы оптимизации нейронных сетей
- Функции активации нейросети

нетология

Спасибо за внимание!

Александр Сибагатов

Open in Colab