ИТМО

Отчет о выполнении итогового проекта

Выполнили: Чунарев Иван

Смирнов Данил

Описание набора данных

Данные описывают исторические поминутные биржевые характеристики фонда IMOEX, которые были собраны с помощью Tinkoff Invest API

Признак	Описание
FIGI	Уникальный номер инструмента
UTC	Время выполнения сделки в формате UTC
open	Цена на момент открытия периода
close	Цена на момент закрытия периода
high	Высшая цена за период
low	Низшая цена за период
volume	Объем торгов инструментом за период

Предварительная обработка данных

```
years = dict()
for i in range(2020,2024):
  years[i] = list()

path = '/content/drive/MyDrive/Train_dataset/'
directory = os.listdir(path)
for dirnum in range(len(directory)):
  curPath = path + directory[dirnum] #путь до директорий с файлами по дням
  curYearDir = os.listdir(curPath)
  for fileName in curYearDir: #пробежаться по каждому файлу папки-года
    curFilePath = curPath + "/" + fileName #путь до файла - дня года
    df = pd.read_csv(curFilePath,
                     header = None,
                     index_col=False,
                     sep = ';',
                     names = ['FIGI', 'UTC', 'open', 'close', 'high', 'low', 'volume']
                     ) #считывание файла
    years[int(directory[dirnum])].append(df)
```

Данные приходят в формате csv, без head-строки, размещенные в папках с номером года.

Создаем словарь, в который производим считывание данных и добавляем заголовочную строку

Предварительная обработка данных

```python
#поиск среднего
def avrg(xarr):
  sum = 0
  num = len(xarr)
  for i in xarr:
    sum += i
  return sum/num

#восстановление значений из нормализованных
def denorm(values, averags):
  for i in range(len(values)):
    values[i] = values[i] * averags[i]
```

Для корректной работы модели все величины нормализуются на среднюю величину 'close' за период seq_len, а после предсказания выполняется денормализация данных

Предварительная обработка данных

```
def gen_xy(day_csv, seq_len):
        input_train = list()
        output_train = list()
        x_avrgs = list()
        for i in range((len(day_csv)//seq_len)*seq_len - seq_len - 1):
            pred_x = day_csv.iloc[i: i + seq_len, 3]
            x_avrg = avrg(pred_x)
            x = np.array(pred_x).__itruediv__(x_avrg)
            y = np.array([day_csv.iloc[i + seq_len + 1, 3]], np.float64)
            .__itruediv__(x_avrg)
            x_avrgs.append(x_avrg)
            input_train.append(x)
            output_train.append(y)
        X_train = np.array(input_train)
        Y_train = np.array(output_train)
        return X_train, Y_train, x_avrgs
```

После чего нормализованные величины сохраняются в массив и возвращаются функцией, генерирующей датасеты, для построения графиков в исходных величинах. Модели на вход будет подаваться информация о ценах закрытии ('close') предыдущих seq_len минут торгов.

Предварительная обработка данных

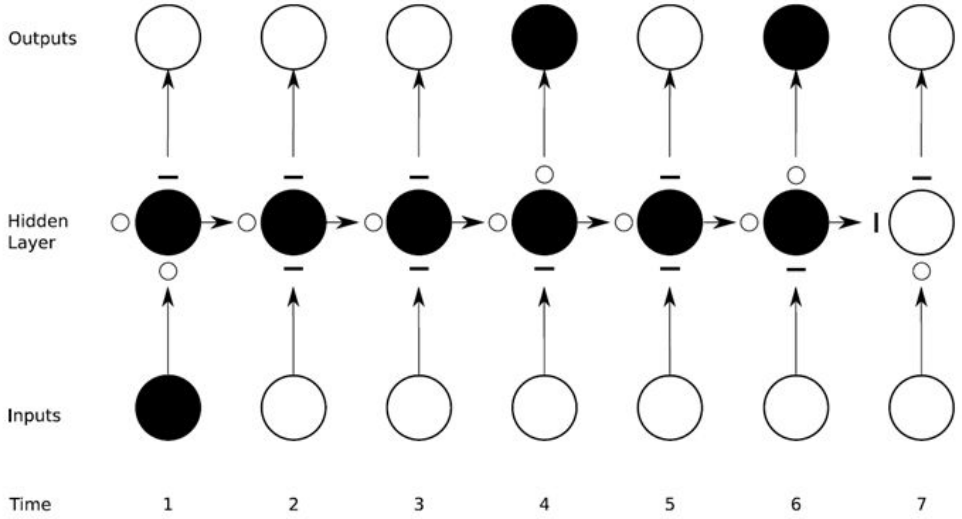
```
def gen_xy_differance(day_csv, seq_len):
        input_train = list()
        output_train = list()
        x_avrgs = list()
        for i in range((len(day_csv)//seq_len)*seq_len - seq_len - 1):
            pred_x = day_csv.iloc[i: i + seq_len, 3]
            x_avrg = avrg(pred_x)
            x = np.array(pred_x).__itruediv__(x_avrg)
            #Нормализованная разница цен
            y = np.array(([day_csv.iloc[i + seq_len + 1, 3] - day_csv.iloc[i + seq_len, 3]]),
                         np.float64).__itruediv__(x_avrg)

            x_avrgs.append(x_avrg)
            input_train.append(x)
            output_train.append(y)
        X_train = np.array(input_train)
        Y_train = np.array(output_train)
        return X_train, Y_train, x_avrgs
```

Мы также пытались предсказывать не саму цену, а разницу цены между текущей минутой и следующей, что имеет большее значение для торговли чем сама цена.

Но к сожалению этот метод показал худшие результаты

Постановка задачи и построение модели

В качестве модели использовалась LSTM - (long shirt-term memory). Она является часто применимым решением при работе с временными рядами, поэтому мы остановились на ней.



Постановка задачи и построение модели

```
predictors = 8

model = tf.keras.Sequential()
model.add(tf.keras.layers.LSTM(20, input_shape=(predictors, 1),
                               return_sequences=True))
model.add(tf.keras.layers.LSTM(20))
model.add(tf.keras.layers.Dense(1, activation=tf.nn.relu))
model.compile(optimizer="adam",
              loss="mean_squared_error",
              metrics = ['accuracy', 'mean_squared_error'])
```

В переменную predictors записывается размер входа модели, то есть сколько предыдущих минут подается на вход для предсказания следующей. В ходе экспериментов мы остановились на 8. Выходной слой имеет один выход - предсказанная нормализованная цена на следующую минуту. В качестве функции потерь использован MSE.

Постановка задачи и построение модели

После чего запускаем обучение модели

```
'''
Обучение на выборке всех дней 2020-2023 годов
'''
for y in range(2020, 2024):
  for i in range(len(years[y])):
    X_train, Y_train, x_avrgs = gen_xy(years[y][i], predictors)
    '''
    В качестве batch_size берется количество торгов за день
    Так сделано, поскольку при новом batch память LSTM модели очищается,
    и для учета всех предыдущих минут дня нужно все минуты дня иметь в одном
    batch
    '''
    model.fit(X_train, Y_train, epochs=8, batch_size = len(x_avrgs))
```

Оценка модели и результатов

Далее выведем полученные результаты на график для оценки успешности оценки

```
#Получение выборки и получение результатов работы модели
X_predict, Y_predict, x_avrgs = gen_xy(years[year][day], predictors)
pred_arr = model.predict(X_predict)
pred = pred_arr.reshape(1, len(pred_arr))[0]
t = np.arange(0, len(pred), 1)

#Денормализация реального и предсказанного выходов
denorm(Y_predict, x_avrgs)
denorm(pred, x_avrgs)

#Построение графика окном в случайные lim_size минут выбранного дня
plt.plot(t, pred, 'r', t, Y_predict, 'g--')
lim_size = 40
lim = random.randint(lim_size, len(t))
plt.xlim(lim - lim_size, lim)
```

Оценка модели и результатов

Исходя из графика можно сделать вывод, что предсказание было выполнено достаточно близко к действительным ценам. Самое важное, что предсказанные данные в большинстве случаев правильно предсказывают рост или падение цены инструмента

Числовые метрики оценки модели:

loss: 1.8881e-06

accuracy: 0.0263

mean_squared_error: 1.8881e-06

Эксперименты

Результаты модели учитывающей не только цену закрытия, но и объем торгов за минуту

Теоретически объем должен влиять на динамику изменения цены

Эксперимент с разными количествами эпох и разным количеством минут, подаваемых на вход

Результаты предсказания модели, которая в качестве предсказываемой величины выдает разницу между текущим значением цены и будущим значением

Спасибо
за внимание!

ITMO more than a UNIVERSITY

Ваши контакты