Факультет компьютерных наук

Машинное обучение и высоконагруженные системы

Москва 2024

Предсказание стоимости акций

Участники: Алексеев Павел
Куратор: Фофанова Татьяна

Описание проекта:

Предсказание стоимости акций с горизонтом 30 дней с помощью моделей ML и DL:

1. Пользователь отправляет тикер (название акции)
2. Пользователь получает прогноз на следующие 30 дней торгов
3. Пользователей получает дополнительно прогнозы от экспертов (из открытых источников, yahoo finance)

Состав команды:

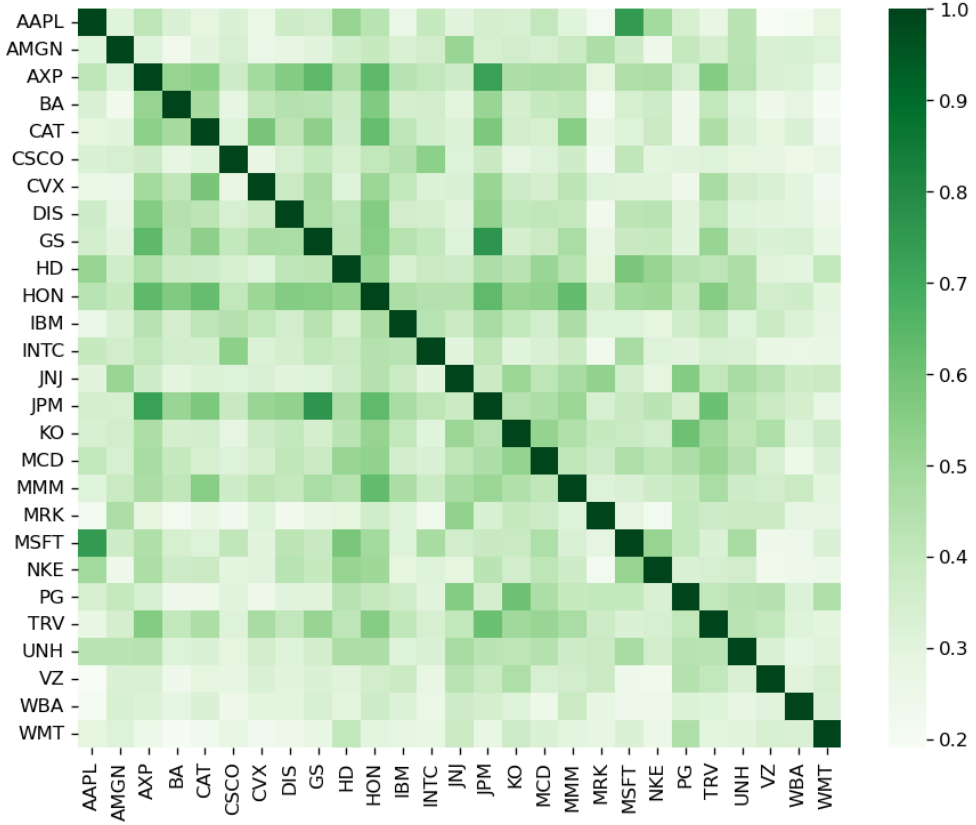
Алексеев Павел

Куратор:

Фофанова Татьяна

Первый этап: EDA

1. Использовались данные компаний индекса Dow Jones
2. Проведен тест на стационарность по данным, очищенным от тренда. Все тикеры его прошли.
3. Проверена гипотеза о равенстве математических ожиданий логдоходностей и гипотеза о равенстве дисперсий логдоходностей. Более 95% результатов – равенство.
4. Проверена гипотеза о нормальном распределении логдоходностей. Гипотеза отклонена для всех акций.
5. Построена матрица корреляций. Большинство сильных корреляций приходится на компании одного сектора. Например The Goldman Sachs и J.P. Morgan, или же Apple и Microsoft. Остальные тикеры коррелируют слабо.



Можем сделать вывод о том, что распределение параметров схоже. То есть мы можем использовать модель, обученную на большом количестве тикеров, для предсказания тех данных, на которых обучение не происходило. Более продвинутая модель – разделенная по секторам, но это на будущее.

Описание данных для ML:

Загружаются данные по 29 тикера из индекса dow jones за период с 2012-01-01 по 2024-01-01.
(1 исключается из-за наличия большого количества пропусков).

Предобработка данных:

1. Генерируются лаговые фичи с помощью рукописной функции. Для каждой комбинации лага, окна и метрики создается новый столбец. Далее удаляются все столбцы для окна 1 за исключение mean, так как всегда являются полностью нулевыми.

```
 lags=[30,45,60,75,90,180,365],
 windows=[1,2,3,4,5,10,20,30,60,90,180,365],
metrics=['mean', 'var', "percentile_90", "percentile_10"]
```

2. Далее добавляется в качестве dummy переменной столбцы с обозначением дней недели.

3. На следующем шаге данные по всем тикерам объединяются в единый датафрейм.

4. Данные разделяются на train и test. Так как данные – временные ряды, shuffle = False

5. На следующем шагу происходит трансформация Бокса-Кокса, для нормализации данных.

Обучение моделей:

Для тестирования использовались модели Градиентного бустинга и случайный лес. Предварительно для каждой из моделей перебирались оптимальные показатели с помощью optuna.

Сравнивались следующие модели:

```
GB_model = lgb.LGBMRegressor(n_estimators=1000, max_depth=-1, random_state=42, n_jobs=10, verbose=-1)
RF_model = RandomForestRegressor(n_estimators=100, random_state=42, n_jobs=10, verbose=0)
```

И линейная комбинация их результатов:

```
y_pred_combined = alpha * y_pred_gb + (1-alpha) * y_pred_rf
```

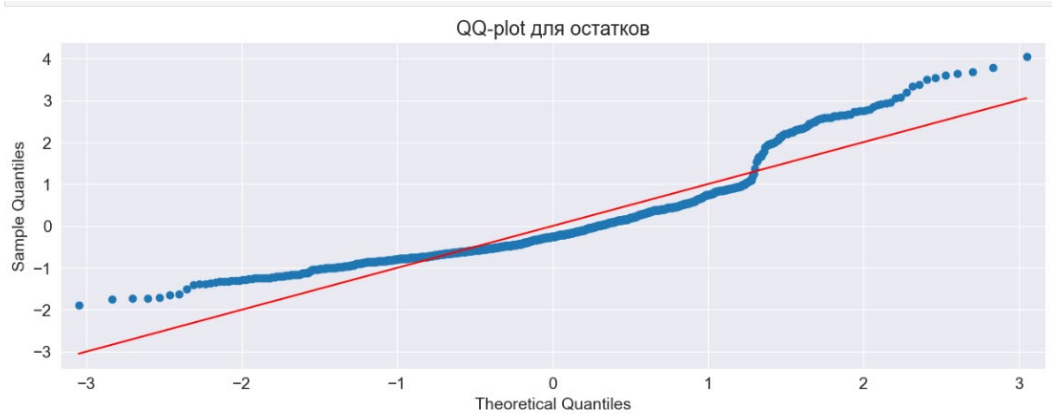
Важный момент. RF с 1000 деревьев показал результат лишь совсем немного лучше, чем с 100 (7.76 у 1000 против 7.8 у 100). Но модель на 1000 деревьев весила 7 гигабайт, поэтому откажемся от нее.

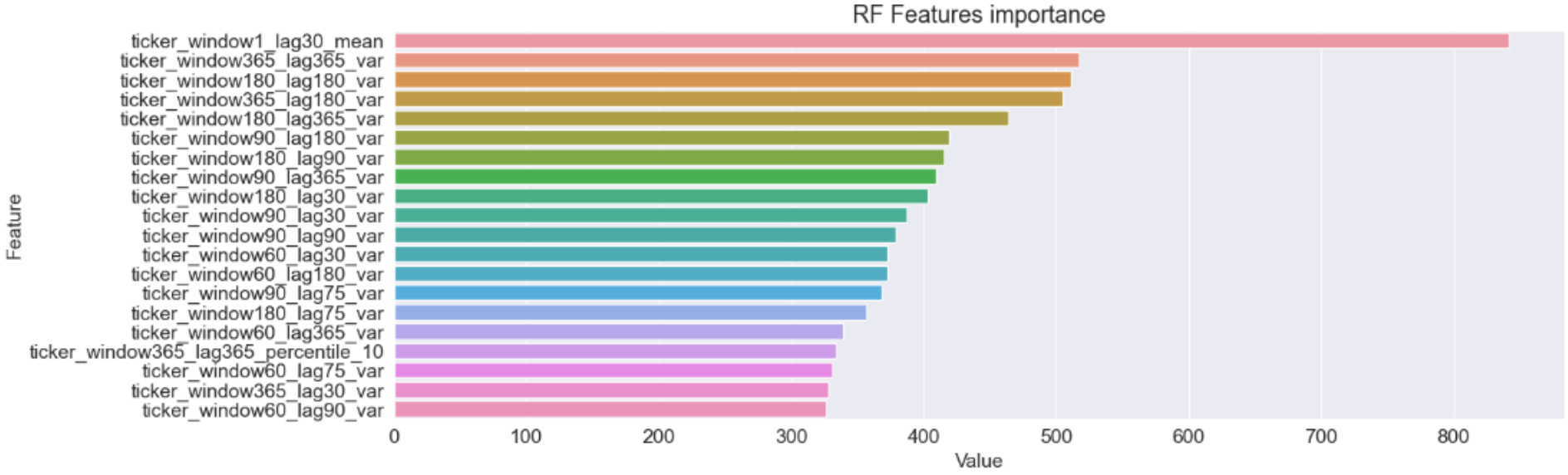
Результаты:

Путем перебора параметра alpha для комбинаций моделей выяснилось, что нет такой линейной комбинации (при alpha != 0), что она будет лучше случайного леса. В таблице покажем значение при alpha = 0.5

	Модель	Время обучения (сек)	Время предсказания (сек)	Оценка MAPE
0	GB_model	15.978247	0.010521	8.272919
1	RF_model	384.784494	0.019000	7.805648
2	Combined_model	400.762741	0.029521	7.992021

Лучший результат на тесте за случайным лесом, эта модель и выбирается в качестве основной. Отобразим самые важные фичи и QQ-plot для остатков





Pipeline (более условный, демонстрирующий работу)

Подаем только тикер, получаем прогноз на следующие 30 дней.

В целях тестирования сделал так, что дата «сегодня» зафиксирована, а мы можем сразу оценить результаты прогноза

```
y_true, y_pred, MAPE = main("AAPL")
print("Предсказания: ", y_pred)
print("Реальные значения: ", y_true)
print("Оценка MAPE: ", MAPE)
```

```
Предсказания:  [184.97003887 182.67861454 181.89571421 181.50878217 182.02784817
 181.79785421 181.45303486 182.02469171 181.79013542 182.02171853
 182.05608222 182.75733572 182.03130889 181.35950804 182.35234912
 181.9687344  182.51192395 182.65169262 181.72482461 181.71057304
 180.76393631 180.22026671 180.55641253 180.69352754 180.51954675
 180.22190004 189.70493599 191.70749246 195.08301977 195.2206323 ]
Реальные значения:  [189.46824646 189.44825745 191.20600891 190.39704895 191.06619263
 189.72790527 189.54812622 190.15734863 189.12866211 189.70791626
 190.99629211 189.18858337 193.17350769 192.07492065 194.02243042
 195.46058655 192.93380737 194.46186829 197.7077179  197.85752869
 197.31822205 195.64035034 196.68902588 194.58171082 194.43188477
 193.35328674 192.8039856  192.90383911 193.33329773 192.28463745]
Оценка MAPE:  5.038539049980053
```

На этом пока все, спасибо за внимание

```python
# Main pipeline
def main(ticker):

    # Просто для теста пусть сегодня - это 30 торговых дней до 1 января 2024 года. Чтобы не лезть в будущее
    ticker = ticker
    start_date='2012-01-01'
    end_date='2024-01-01'
    test_data = pd.DataFrame(yf.download(ticker, start_date, end_date, progress=False)["Adj Close"]).reset_index(drop=False)
    test_data= test_data.rename(columns={"Adj Close": "ticker"})

    #Уберем последние 30 значения. Их мы будем предсказывать. На них же проверим результат
    y_true = test_data[-30:]["ticker"].copy().values
    test_data.loc[test_data.index[-30:], 'ticker'] = np.nan

    full_data = generate_lagged_features(test_data, target_cols=["ticker"],
                                         lags=[30,45,60,75,90,180,365],
                                         windows=[1,2,3,4,5,10,20,30,60,90,180,365],
                                        metrics=['mean', 'var', "percentile_90", "percentile_10"])

    #Добавляем дни недели
    full_data['Date'] = pd.to_datetime(full_data['Date'])
    full_data.loc[:, 'day_of_week'] = full_data['Date'].dt.day_name()
    full_data = pd.get_dummies(full_data,columns=["day_of_week"],drop_first=True)
    #Удаляем полностью пустые столбцы (так как окно 1, не может быть дисперсии и инф. персентиля)
    columns_to_drop = ['ticker_window1_lag30_var', 'ticker_window1_lag45_var', 'ticker_window1_lag60_var',
                       'ticker_window1_lag75_var', 'ticker_window1_lag90_var', 'ticker_window1_lag180_var',
                       'ticker_window1_lag365_var', 'ticker_window1_lag30_percentile_90', 'ticker_window1_lag45_percentile_90',
                       'ticker_window1_lag60_percentile_90', 'ticker_window1_lag75_percentile_90', 'ticker_window1_lag90_percentile_90',
                       'ticker_window1_lag180_percentile_90', 'ticker_window1_lag365_percentile_90', 'ticker_window1_lag30_percentile_10',
                       'ticker_window1_lag45_percentile_10', 'ticker_window1_lag60_percentile_10', 'ticker_window1_lag75_percentile_10',
                       'ticker_window1_lag90_percentile_10', 'ticker_window1_lag180_percentile_10', 'ticker_window1_lag365_percentile_10']
    full_data = full_data.drop(columns=columns_to_drop)

    X = full_data[-30:].drop(columns=["ticker","Date"])

    with open('RF_model.pkl', 'rb') as file:
        loaded_RF_model = pickle.load(file)

    with open('lambda_value.pkl', 'rb') as file:
        loaded_lambda_value = pickle.load(file)

    # Прогноз с использованием pipeline
    forecast = loaded_RF_model.predict(X)
    y_pred = inv_boxcox(forecast, loaded_lambda_value)
    MAPE = mean_absolute_percentage_error(y_true, y_pred)

    return y_true, y_pred, MAPE
```


ВЫСШАЯ • ШКОЛА • ЭКОНОМИКИ •