
+1000%
деняг
TensorFlow

Список технологий:

Код

```python
def build_rnn_model(input_shape):
    """Builds an LSTM-based RNN model."""
    model = Sequential()
    model.add(LSTM(units=50, return_sequences=True, input_shape=input_shape))  # First LSTM layer
    model.add(LSTM(units=100, return_sequences=True))  # Second LSTM layer
    model.add(Dropout(0.2))
    model.add(LSTM(units=100, return_sequences=True))  # Third LSTM layer
    model.add(Dropout(0.2))
    model.add(LSTM(units=100, return_sequences=True))  # Fourth LSTM layer
    model.add(Dropout(0.2))
    model.add(LSTM(units=50, return_sequences=False))  # Fifth LSTM layer
    model.add(Dropout(0.1))
    model.add(Dense(units=1))  # Output layer
    return model
```

https://github.com/zmeyka3310/gamblingai

Код

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler

def load_and_preprocess_data(csv_path, column_name):
    df = pd.read_csv(csv_path)
    try:
        df['Date'] = pd.to_datetime(df['Date'], format='%m/%d/%Y')
    except:
        df['Date'] = pd.to_datetime(df['Date'], format='%Y-%m-%d')
    df = df.sort_values('Date')
    df = df.set_index('Date')
    df[column_name] = df[column_name].str.replace('$', '').astype(float)
    data = df[column_name].values.reshape(-1, 1)
    return data, df
```

```python
def create_sequences(data, seq_length):
    X, y = [], []
    for i in range(len(data) - seq_length):
        X.append(data[i:(i + seq_length)])
        y.append(data[i + seq_length])
    return np.array(X), np.array(y)

def scale_data(data):
    scaler = MinMaxScaler()
    scaled_data = scaler.fit_transform(data)
    return scaled_data, scaler
```

https://github.com/zmeyka3310/gamblingai

Код

```python
def main(CSV_FILE_PATH):
    """Main function to load, preprocess, train, and predict."""
    data, df = preprocessing.load_and_preprocess_data(CSV_FILE_PATH, PRICE_COLUMN)
    # Scale the data
    data, scaler = preprocessing.scale_data(data)
    # Create sequences
    X, y = preprocessing.create_sequences(data, SEQUENCE_LENGTH)
    # Split into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=TEST_SIZE, shuffle=False)
    # Build the RNN model
    model = build_rnn_model((SEQUENCE_LENGTH, 1))
    # Train the model
    model = train_model(model, X_train, y_train, EPOCHS, BATCH_SIZE, LEARNING_RATE)
    # Get the last sequence from the original scaled data
    last_sequence = data[-SEQUENCE_LENGTH:]
    # Predict the next value
    predicted_price = predict_next_value(model, last_sequence, scaler)
    # Get the last date from the original dataframe
    last_date = df.index[-1]
    # Calculate the next day
    next_date = last_date + pd.Timedelta(days=1)
    print(f"Predicted price for {next_date.strftime('%Y-%m-%d')}: ${predicted_price:.2f}")
    return predicted_price
```

https://github.com/zmeyka3310/gamblingai

Код

```
def batcher(CSV_FILE_PATH):
        batches = 10
        predicts = []

        for i in range(batches):
                print("----------------------------------------")
                print("")
                print("")
                print(f"Doing batch {i+1}/{batches}")
                print("")
                print("")
                print("----------------------------------------")
                predicts.append(main(CSV_FILE_PATH))

        print()
        print("----------------------------------------")
        print()

        for item in predicts:
                print(f"Predicted price: ${item:.2f}")



        print(f"Average predicted price: ${sum(predicts)/len(predicts)}")
        return sum(predicts)/len(predicts)

if __name__ == "__main__":
        batcher("historicaldata/HDnvda5y.csv")
```

https://github.com/zmeyka3310/gamblingai

Код

```
import os
from batcher import batcher

def process_files(directory):
    file_data = []
    for filename in os.listdir(directory):
        filepath = os.path.join(directory, filename)
        if os.path.isfile(filepath):
            result = batcher(filepath)
            file_data.append((filename, result))

    for filename, result in file_data:
        print(f"File: {filename}")
        print(f"Result: {result}")

if __name__ == "__main__":
    current_directory = "HD5ytesting_nvda"
    process_files(current_directory)
```

https://github.com/zmeyka3310/gamblingai

Код

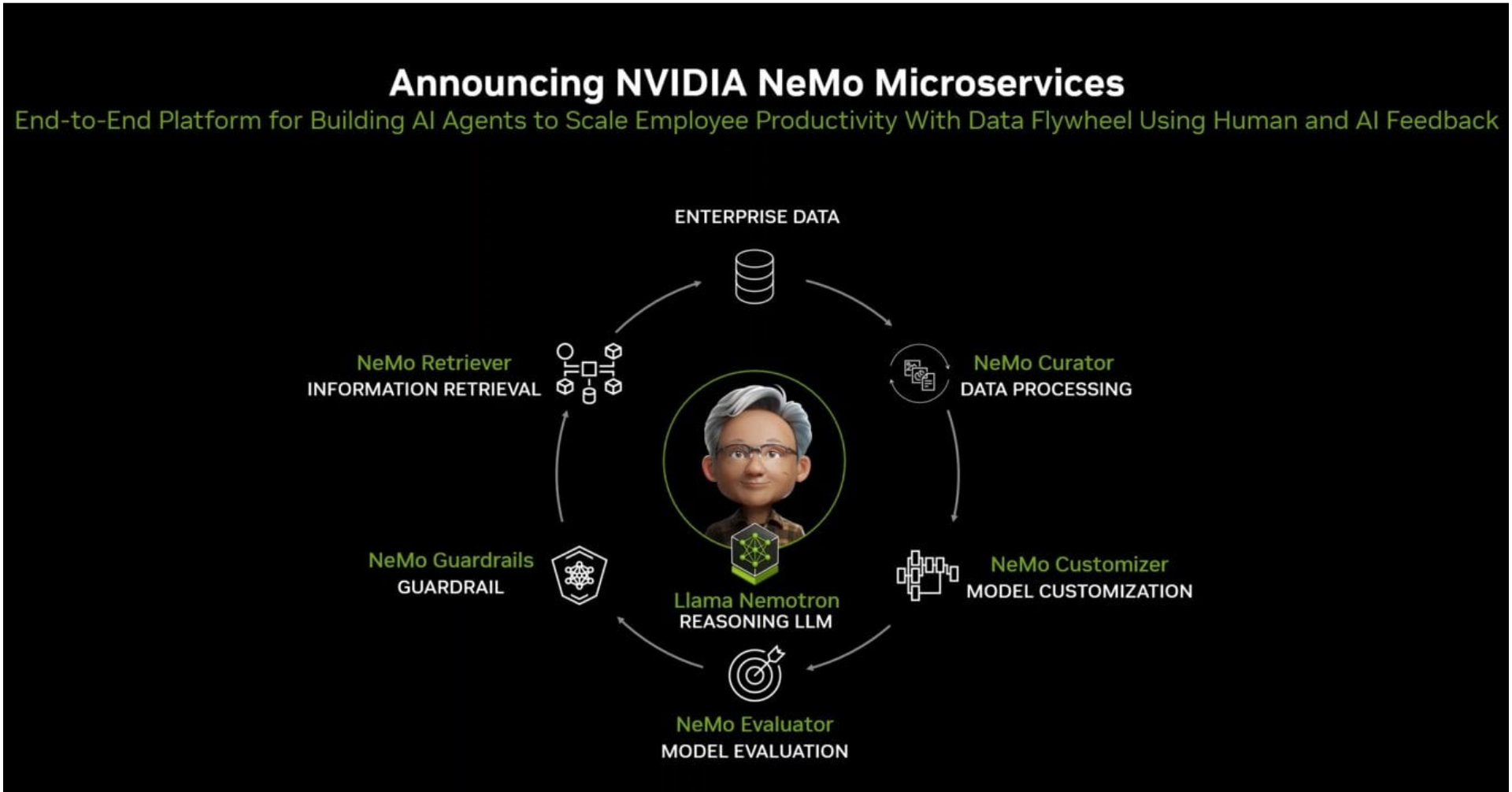
```
# Read data into DataFrames
predicted_df = pd.read_csv(StringIO(predicted_data), sep='\s+', names=['Date', 'Predicted_Price'])
actual_df = pd.read_csv(StringIO(actual_data), sep='\s+', names=['Date', 'Actual_Price'])

# Convert 'Date' to datetime objects
predicted_df['Date'] = pd.to_datetime(predicted_df['Date'], format='%m/%d/%Y')
actual_df['Date'] = pd.to_datetime(actual_df['Date'], format='%m/%d/%Y')

# Sort DataFrames by date (important for calculating returns)
predicted_df = predicted_df.sort_values(by='Date').reset_index(drop=True)
actual_df = actual_df.sort_values(by='Date').reset_index(drop=True)

# Merge the DataFrames based on 'Date'
df = pd.merge(predicted_df, actual_df, on='Date', how='inner')

# Calculate Returns
df['Old_Price'] = df['Actual_Price'].shift(1)  # Get the previous day's actual price
df = df.dropna()  # remove the first row which will have a NaN value for Old_Price

df['Actual_Return'] = (df['Actual_Price'] / df['Old_Price'] - 1.0)
df['Forecast_Return'] = (df['Predicted_Price'] / df['Old_Price'] - 1.0)

# Define the Universe (Let's assume all days are in the universe)
df['Universe'] = True  # All stocks are in the universe
df['Weight_Simple'] = df['Universe'].astype(int) # weight is 1 if in universe, 0 otherwise
```

https://github.com/zmeyka3310/gamblingai

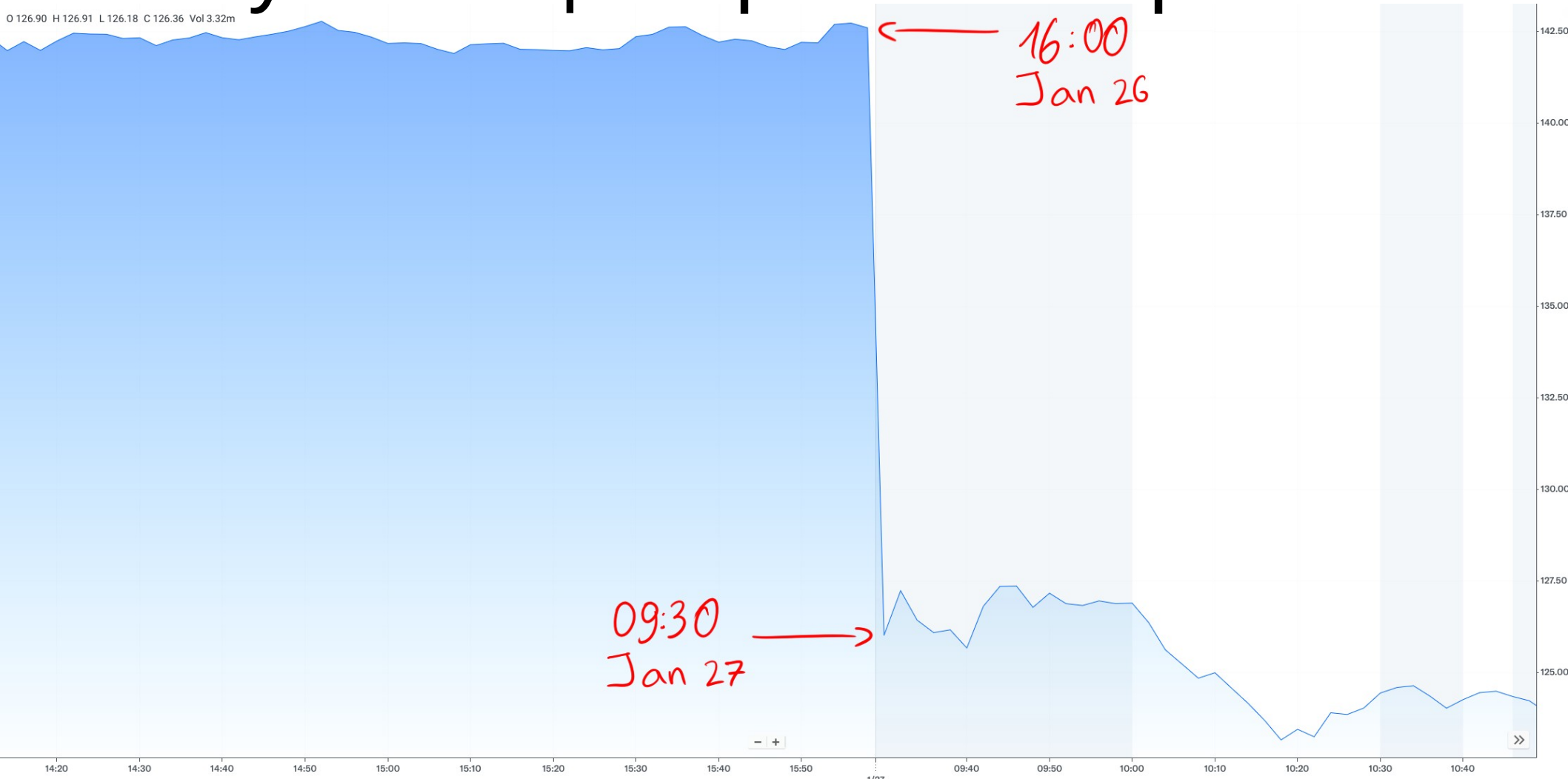
Анализ данных nvda за апрель 2025

	Date	Predicted_Price	Actual_Price	Old_Price	Actual_Return	Forecast_Return	Universe	Weight_Simple
1	2025-04-02	107.290	120.654984	121.001343	-0.002862	-0.113316	True	1
2	2025-04-03	103.510	120.457031	120.654984	-0.001641	-0.142099	True	1
3	2025-04-04	98.910	117.120934	120.457031	-0.027695	-0.178877	True	1
4	2025-04-07	87.460	118.441605	117.120934	0.011276	-0.253250	True	1
5	2025-04-08	103.805	114.176285	118.441605	-0.036012	-0.123577	True	1
6	2025-04-09	98.890	112.904823	114.176285	-0.011136	-0.133883	True	1
7	2025-04-10	109.370	110.218651	112.904823	-0.023791	-0.031308	True	1
8	2025-04-11	108.500	107.596191	110.218651	-0.023793	-0.015593	True	1
9	2025-04-14	114.110	106.887497	107.596191	-0.006587	0.060539	True	1
10	2025-04-15	110.970	104.247292	106.887497	-0.024701	0.038194	True	1
11	2025-04-16	104.550	107.070778	104.247292	0.027085	0.002904	True	1
12	2025-04-17	104.450	105.490906	107.070778	-0.014755	-0.024477	True	1
13	2025-04-21	98.770	106.665878	105.490906	0.011138	-0.063711	True	1
14	2025-04-22	98.780	105.985756	106.665878	-0.006376	-0.073931	True	1
15	2025-04-23	104.520	109.672325	105.985756	0.034784	-0.013830	True	1
16	2025-04-24	103.475	111.707169	109.672325	0.018554	-0.056508	True	1
17	2025-04-25	106.850	109.782875	111.707169	-0.017226	-0.043481	True	1
18	2025-04-28	109.690	111.169472	109.782875	0.012630	-0.000846	True	1
19	2025-04-29	107.670	109.307678	111.169472	-0.016747	-0.031479	True	1
20	2025-04-30	104.470	108.387070	109.307678	-0.008422	-0.044257	True	1

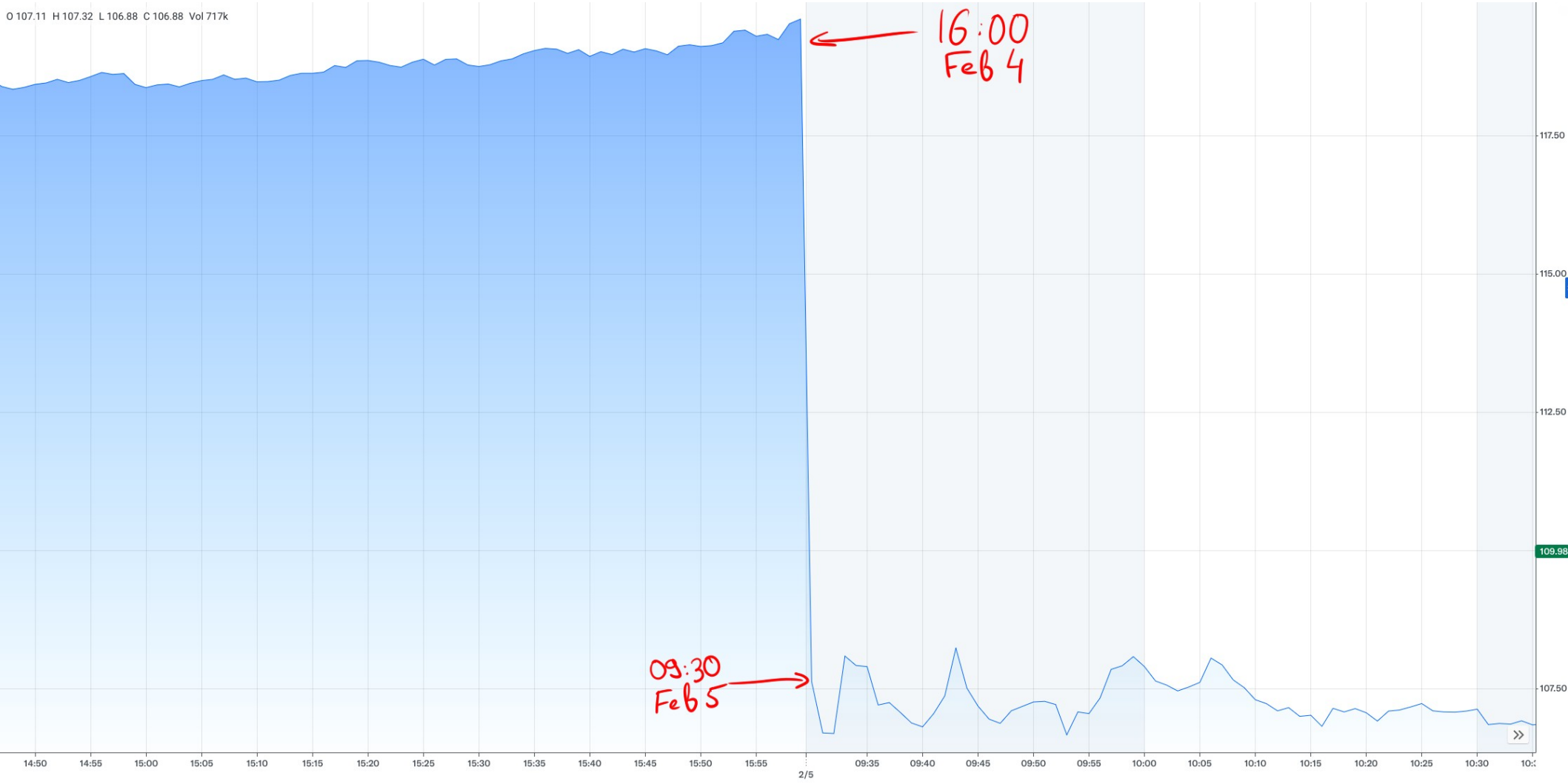
И где мы просчитались?



Публикация трат от deepseek



Частичный отказ amd от ИИ



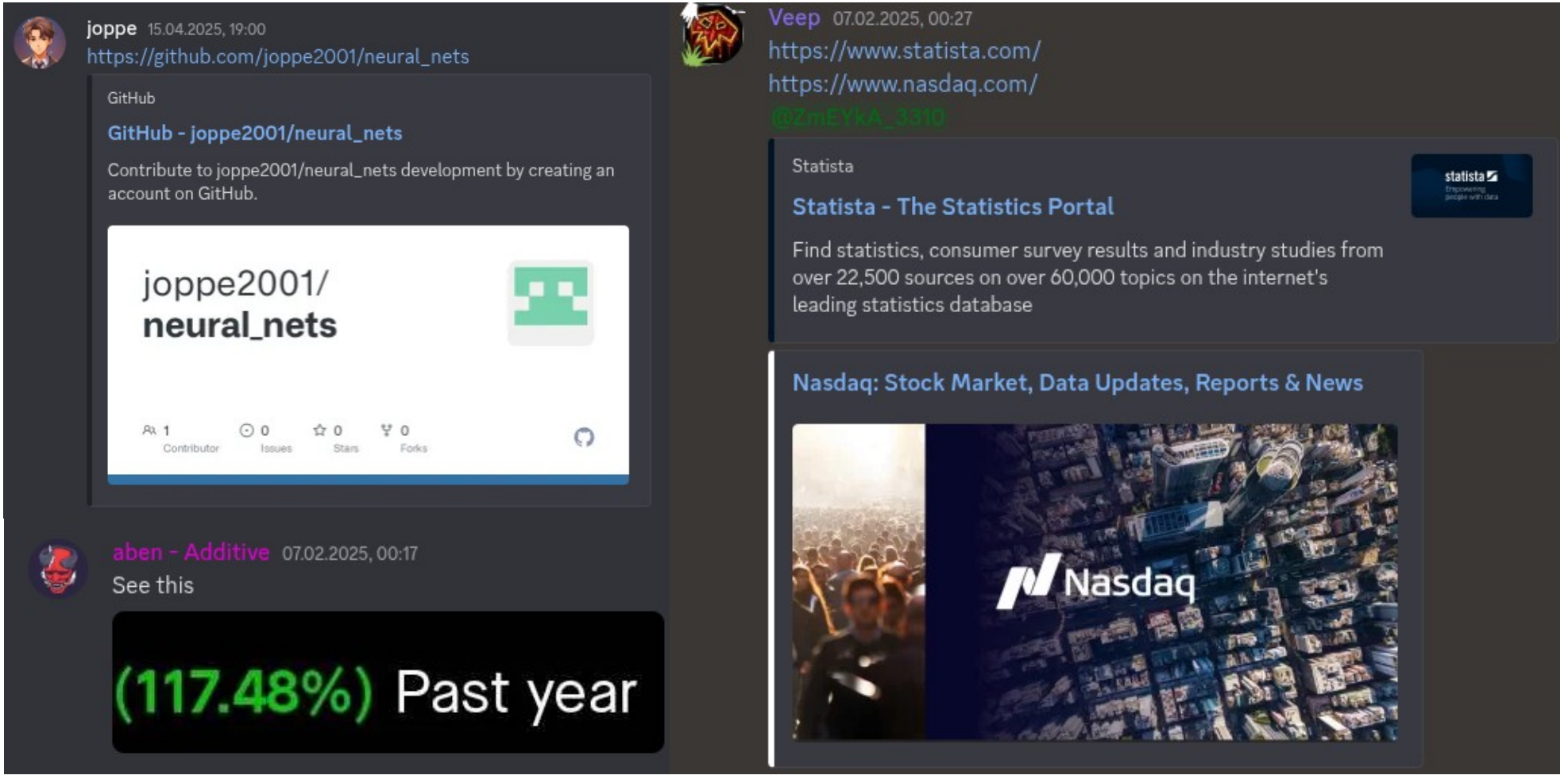
Потенциальные проблемы

- Их нет

Потенциальные решения

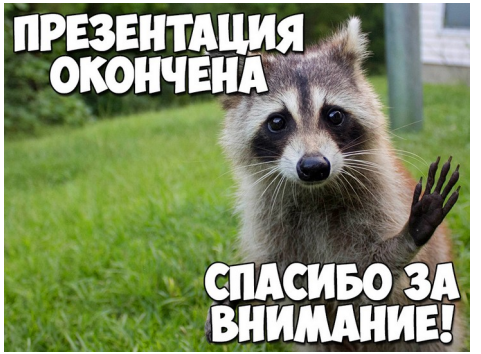
- Использовать всю информацию в файлах, а не только столбец который пытаемся предугадать.
- Больше циклов тренировки модели
- Использовать информацию из внешних источников
- Тренировать все модели параллельно.
- ????

Выражаю благодарность этим людям:



Спасибо за внимание

Спасибо за внимание



Спасибо за внимание