Project Report: Transformer-Based Trading Model

Model Selection:

Pretrained Transformer (**BERT**): I started with a pretrained BERT model for sequence classification. The model is fine-tuned in provided dataset to predict the trading signals. BERT was chosen for its ability to handle classification tasks with strong performance.

Feature Engineering(Used Existing momentum provided in a PPO Code)

1. Selection of Technical Indicators:

- Close Price (`**Close**`): The closing price of the asset.
- Relative Strength Index (`**RSI**`): A momentum indicator that measures the magnitude of recent price changes.
- Moving Average Convergence Divergence (`**MACD**`): A trend-following momentum indicator.
- Stochastic Oscillator K (`**Stoch_k**`): A momentum indicator comparing a particular closing price to a range of its prices over time.
- Average True Range (`**ATR**`): Used to measure volatility.
- Other Indicators: Indicators such as **OBV**, Bollinger Bands, ADX, and others were also considered to capture different market conditions.

2. Textual Representation of Features:

The indicators were combined into a textual format that could be tokenized and fed into the transformer model. Example:

```
text = (
    f"Price is {row['Close']}, RSI is {row['RSI']}, MACD is {row['MACD']}, "
    f"Stochastic K is {row['Stoch_k']}, ATR(14) is {row['ATR_14']}, "
    f"ADX is {row['ADX']}"
)
```

Hyperparameter Tuning:

1. Initial Setup:

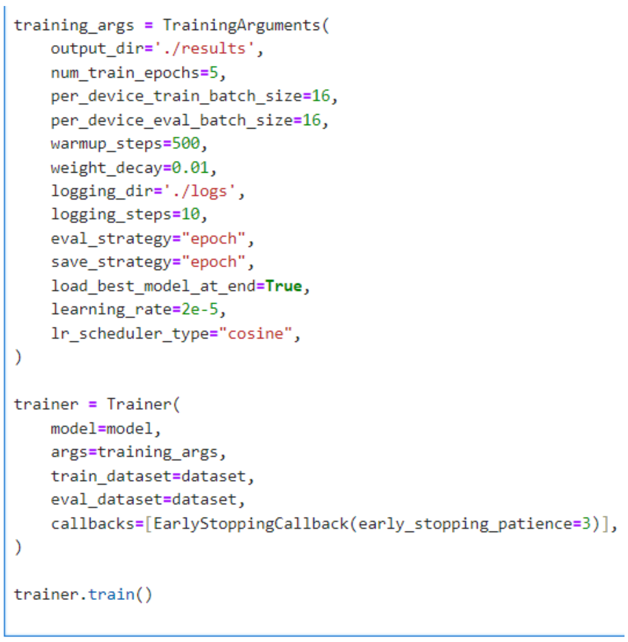
- **Learning Rate**: Started with a standard learning rate of `**2e-5**`, typical for fine-tuning transformer models.

- Batch Size: Set to `16` to balance between computational efficiency and model convergence.

- Epochs: Initially set to `5` epochs to observe the model's learning curve, with early stopping configured to prevent overfitting.

2. **Learning Rate Scheduling(cosine and linear):**

Cosine

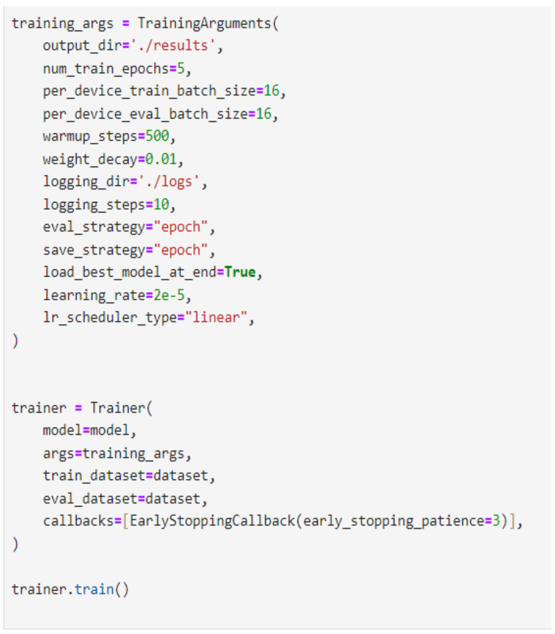


```
training_args = TrainingArguments(
    output_dir='./results',
    num_train_epochs=5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    warmup_steps=500,
    weight_decay=0.01,
    logging_dir='./logs',
    logging_steps=10,
    eval_strategy="epoch",
    save_strategy="epoch",
    load_best_model_at_end=True,
    learning_rate=2e-5,
    lr_scheduler_type="cosine",
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=dataset,
    eval_dataset=dataset,
    callbacks=[EarlyStoppingCallback(early_stopping_patience=3)],
)

trainer.train()
```

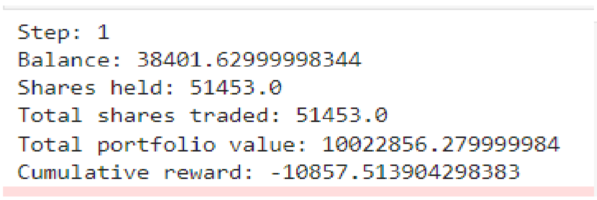
Linear



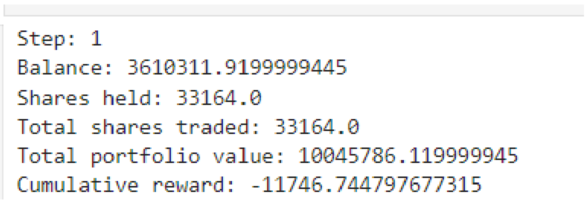
```
training_args = TrainingArguments(
    output_dir='./results',
    num_train_epochs=5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    warmup_steps=500,
    weight_decay=0.01,
    logging_dir='./logs',
    logging_steps=10,
    eval_strategy="epoch",
    save_strategy="epoch",
    load_best_model_at_end=True,
    learning_rate=2e-5,
    lr_scheduler_type="linear",
)


trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=dataset,
    eval_dataset=dataset,
    callbacks=[EarlyStoppingCallback(early_stopping_patience=3)],
)

trainer.train()
```



```
Step: 1
Balance: 38401.62999998344
Shares held: 51453.0
Total shares traded: 51453.0
Total portfolio value: 10022856.279999984
Cumulative reward: -10857.513904298383
```



```
Step: 1
Balance: 3610311.9199999445
Shares held: 33164.0
Total shares traded: 33164.0
Total portfolio value: 10045786.119999945
Cumulative reward: -11746.744797677315
```

- Implemented a linear learning rate scheduler with warmup to gradually increase the learning rate at the beginning of training and then decrease it, helping the model converge more smoothly.
- Warmup Steps: Initially set to `500` to allow the model to start with smaller updates.

4. **Early Stopping:**

- Early Stopping: Configured with a patience of `3`, monitoring the validation loss to halt training if no improvement was observed, thus saving time and resources.

Integrate with PPO Model:

- **Environment Setup**: The trading environment (**TradingEnvironmentWithTransformer**) was customized to incorporate both the Transformer model's predictions and the PPO model's actions.
- **_take_action Method**:
    - Transformer model was used to predict the trading signal (Buy, Sell, Hold).
    - The PPO model received these predictions and other state variables as inputs to make a final decision.

Model Performance:

- Training Loss: Final training loss settled at 0.240900.
- Validation Loss: Final validation loss improved to 0.225595, indicating good generalization performance.