feature_selection

May 10, 2024

1 Feature Selection

[ ]:
```
import sys

sys.path.insert(1, "/Users/simon/Documents/II/Dissertation/")
%load_ext autoreload
%autoreload 2
from src.misc import load_processed_dataset
import torch
import yaml
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

[ ]:
```
df = load_processed_dataset("aapl")
X = df.drop([x for x in df.columns if "Forecast" in x], axis=1)
y = df["Close"]
```

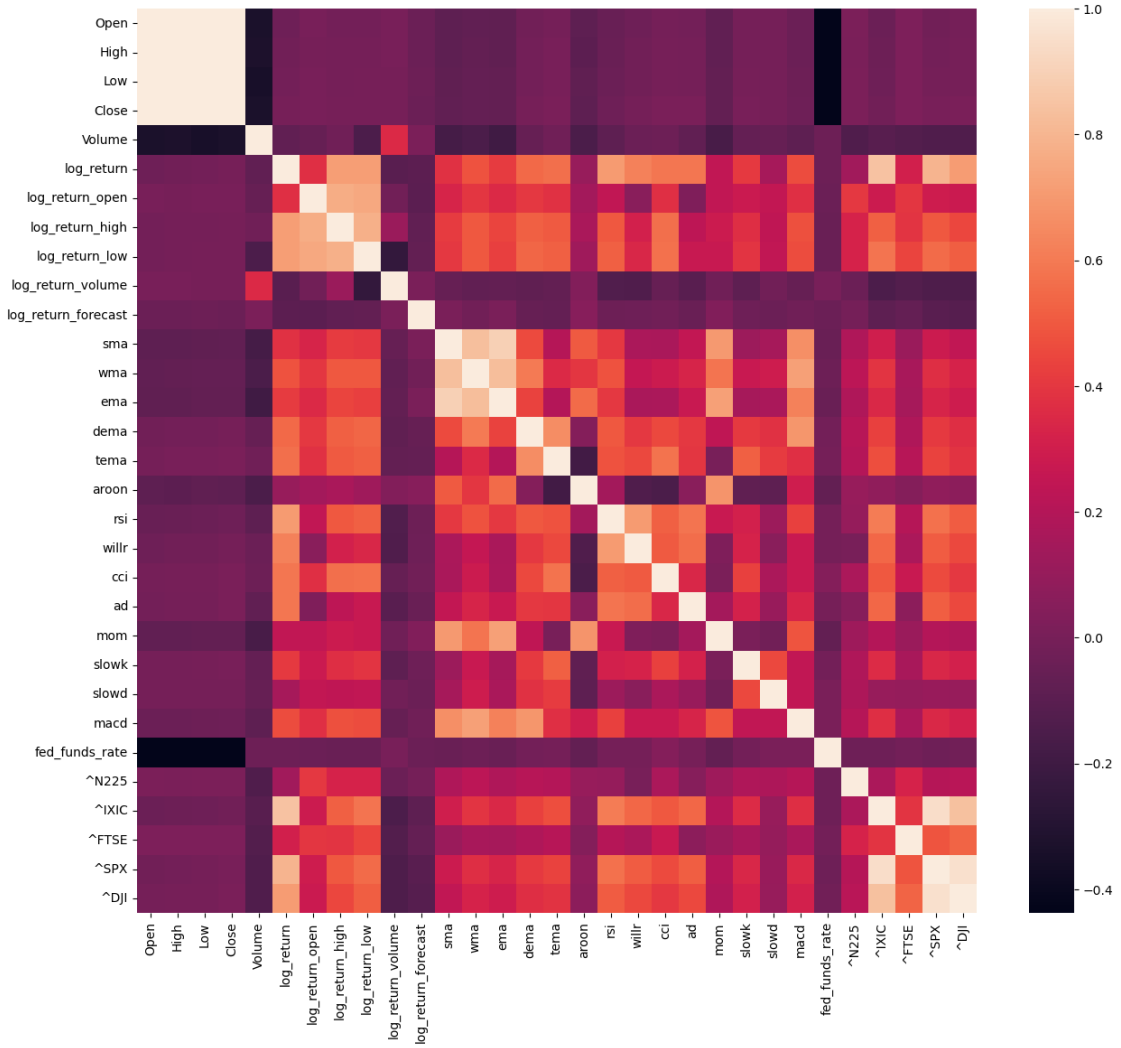
1.1 Filter Methods

Correlation Coefficient

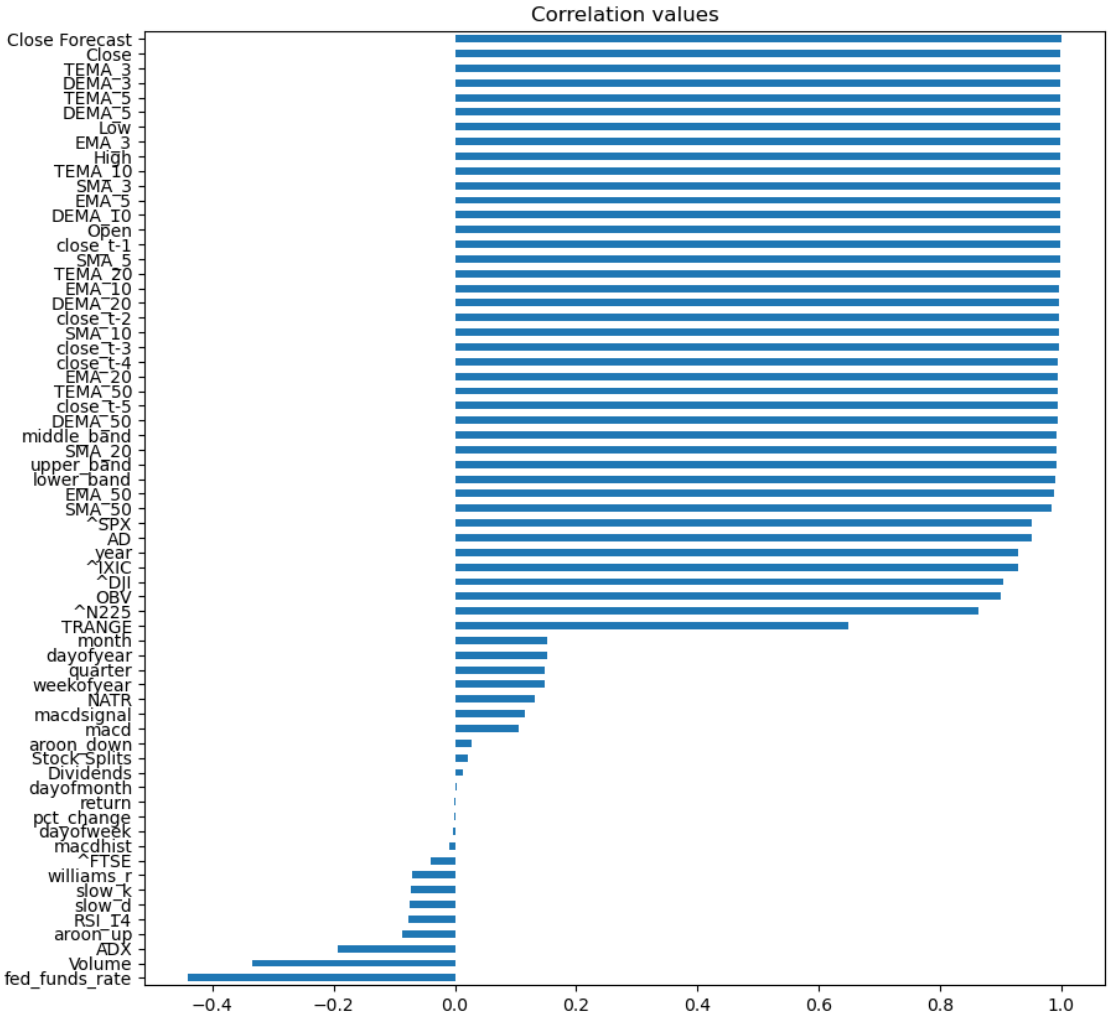
We won't use this in aggregate score because only detects linear relationships.

[ ]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

correlation_matrix = df.corr()

plt.figure(figsize=(15, 13))  # Increase figure size
sns.heatmap(correlation_matrix, fmt=".2g")
```

[ ]:
```
<Axes: >
```

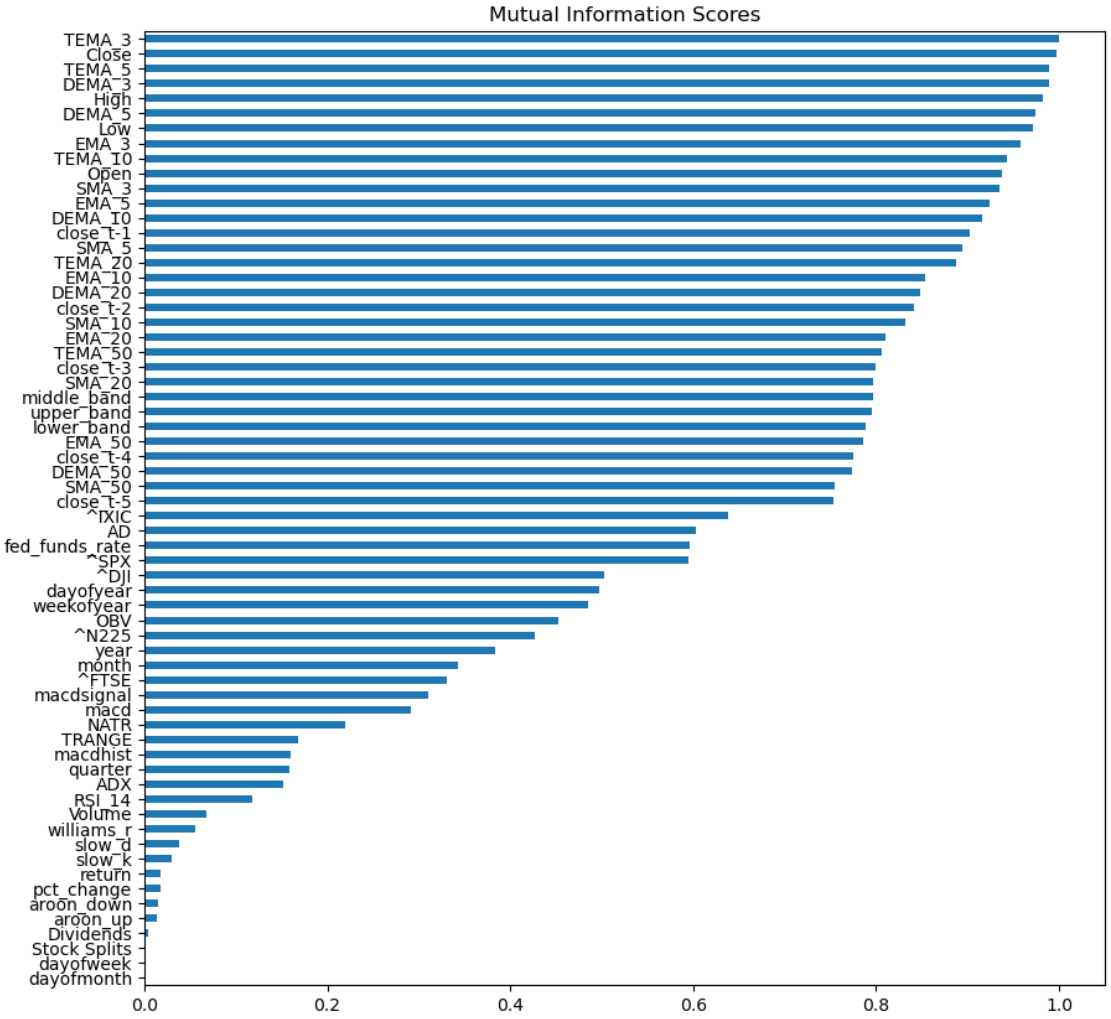


[ ]:
```
corr_values = correlation_matrix["Close Forecast"].sort_values(ascending=True)
corr_values.plot.barh(figsize=(10, 10))
plt.title("Correlation values")
plt.show()
```

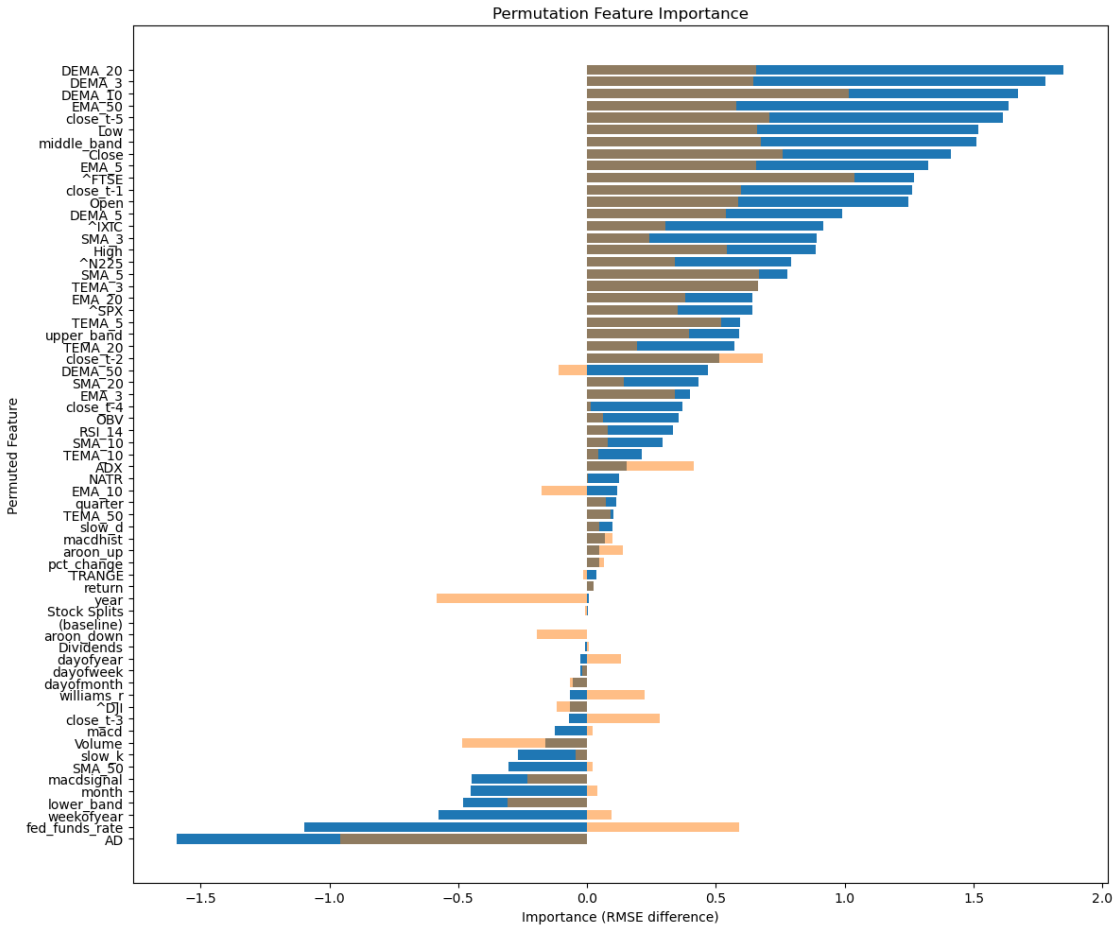


Mutual Information

[ ]:
```
from sklearn.feature_selection import mutual_info_regression
import pandas as pd
import numpy as np

mi_scores = mutual_info_regression(X, y, random_state=42)
mi_scores = pd.Series(mi_scores, name="MI Scores", index=X.columns)
mi_scores = mi_scores.sort_values(ascending=True)  # Sorting the scores
mi_scores = (mi_scores - min(mi_scores)) / (
    max(mi_scores) - min(mi_scores)
)  # Normalise
mi_scores.plot.barh(figsize=(10, 10))
plt.title("Mutual Information Scores")
plt.show()
```



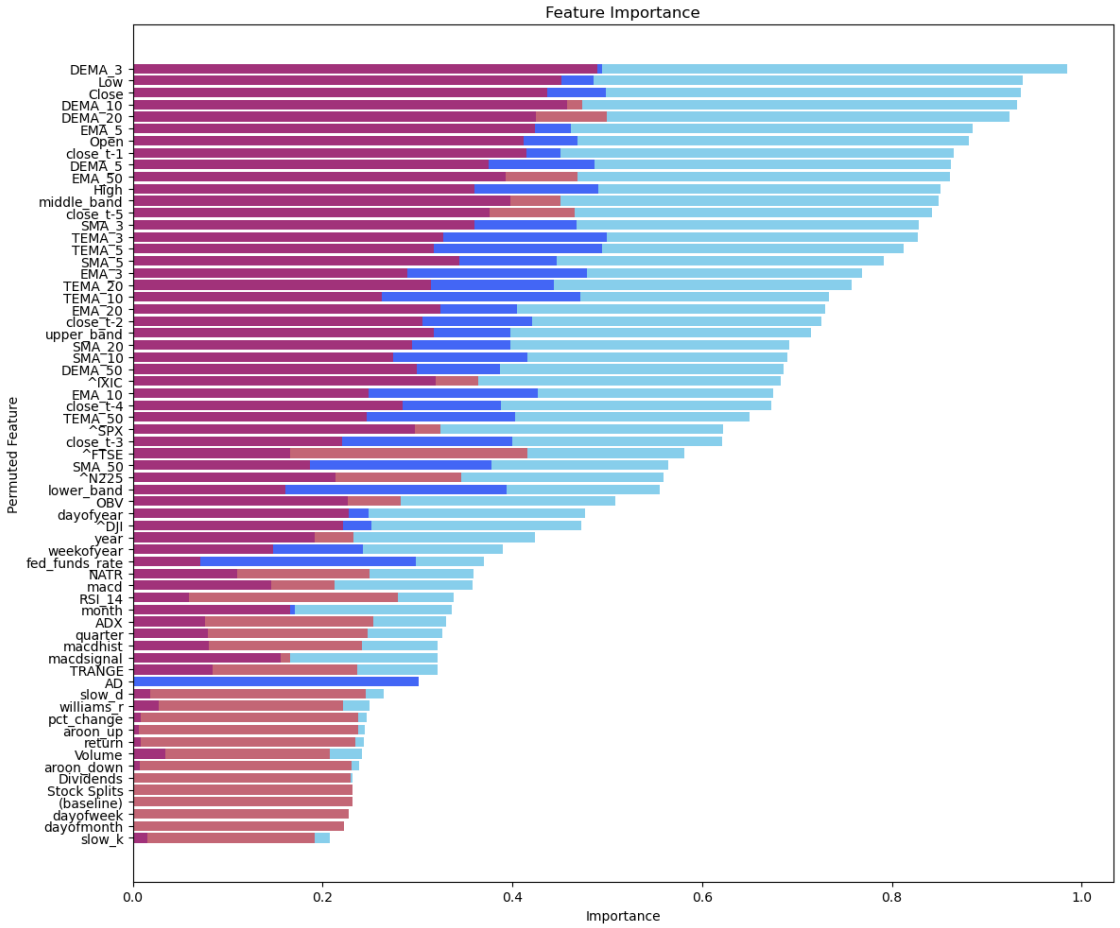
[ ]:
```
pfi_scores = pfi_scores.sort_values(by="test_importance", ascending=True)
plt.figure(figsize=(12, 10))
plt.barh(pfi_scores["feature"], pfi_scores["test_importance"])
plt.barh(pfi_scores["feature"], pfi_scores["val_importance"], alpha=0.5)
plt.xlabel("Importance (RMSE difference)")
plt.ylabel("Permuted Feature")
plt.title("Permutation Feature Importance")
plt.tight_layout()
plt.show()
```



Combining MI and PFI

[ ]:
```
pfi_scores = pd.Series(
    data=pfi_scores["test_importance"].values, index=pfi_scores["feature"]
)
pi_scores_norm = (pfi_scores - min(pfi_scores)) / (max(pfi_scores) -␣
 ↪min(pfi_scores))
comb_scores = (pi_scores_norm / 2).add(mi_scores / 2, fill_value=0)
comb_scores = comb_scores.sort_values()
# Plotting
plt.figure(figsize=(12, 10))
plt.barh(comb_scores.index, comb_scores.values, color="skyblue")
plt.barh(mi_scores.index, mi_scores.values / 2, alpha=0.5, color="blue")
plt.barh(pi_scores_norm.index, pi_scores_norm.values / 2, alpha=0.5,␣
 ↪color="red")
plt.xlabel("Importance")
plt.ylabel("Permuted Feature")
plt.title("Feature Importance")
```

```
plt.tight_layout()
plt.show()
```



Evaluating all feature sets

[ ]:
```
feature_sets = {
    "ohlcv": ["Open", "High", "Low", "Close", "Volume"],
    "all": feature_set,
}

ks = [10, 30, 50]
for k in ks:
    feature_sets[f"top_{k}"] = comb_scores.nlargest(k).index.tolist()
```

[ ]:
```
with open("../configs/lstm_tune.yaml", "r") as file:
    config = yaml.safe_load(file)

fset_scores = []
```

```
for f_set_name, features in (pbar := tqdm(feature_sets.items())):
    pbar.set_description(f"Fitting model to feature set: {f_set_name}")
    cur_config = config.copy()
    cur_config["data"]["init_args"]["feature_set"] = features
    cli = BaseCLI(
        run=False,
        parser_kwargs={"default_config_files": ["../configs/cli_defaults.
 ↪yaml"]},
        args=config,
    )
    cli.trainer.fit(model=cli.model, datamodule=cli.datamodule)
    val_metrics = cli.trainer.validate(
        model=cli.model,
        datamodule=cli.datamodule,
        ckpt_path="best",
        verbose=False,
    )[0]
    test_metrics = cli.trainer.test(
        model=cli.model,
        datamodule=cli.datamodule,
        ckpt_path="best",
        verbose=False,
    )[0]

    fset_scores.append(
        {
            "feature_set": f_set_name,
            "val_rmse": val_metrics["val_rmse"],
            "test_rmse": test_metrics["test_rmse"],
        }
    )

fset_scores = pd.DataFrame(fset_scores)
```

```
Fitting model to feature set: ohlcv:   0%|          | 0/5 [00:00<?,
?it/s]/Users/simon/anaconda3/envs/proj/lib/python3.9/site-
packages/lightning/pytorch/cli.py:520: LightningCLI's args parameter is intended
to run from within Python like if it were from the command line. To prevent
mistakes it is not recommended to provide both args and command line arguments,
got: sys.argv[1:]=['--
f=/Users/simon/Library/Jupyter/runtime/kernel-v2-6677mlgr84Kb8V46.json'],
args={'experiment_name': 'lstm_brk-b_all', 'model': {'class_path':
'src.models.LSTM_Lightning.model.StockPricePredictor', 'init_args':
{'sequence_len': 30, 'hidden_dim': 64, 'num_layers': 1, 'lr': 0.001, 'dropout':
0}}, 'data': {'class_path': 'src.models.LSTM_Lightning.data.StockDataModule',
'init_args': {'stock': 'aapl', 'feature_set': ['Open', 'High', 'Low', 'Close',
'Volume'], 'sequence_len': 30, 'batch_size': 16, 'permute_column': None,
```

```
'dataset_len': 5, 'horizon': 5}}}.
Seed set to 42
Fitting model to feature set: all:  20%|          | 1/5 [00:31<02:06, 31.72s/it]
/Users/simon/anaconda3/envs/proj/lib/python3.9/site-
packages/lightning/pytorch/cli.py:520: LightningCLI's args parameter is intended
to run from within Python like if it were from the command line. To prevent
mistakes it is not recommended to provide both args and command line arguments,
got: sys.argv[1:]=['--
f=/Users/simon/Library/Jupyter/runtime/kernel-v2-6677mlgr84Kb8V46.json'],
args={'experiment_name': 'lstm_brk-b_all', 'model': {'class_path':
'src.models.LSTM_Lightning.model.StockPricePredictor', 'init_args':
{'sequence_len': 30, 'hidden_dim': 64, 'num_layers': 1, 'lr': 0.001, 'dropout':
0}}, 'data': {'class_path': 'src.models.LSTM_Lightning.data.StockDataModule',
'init_args': {'stock': 'aapl', 'feature_set': ['Open', 'High', 'Low', 'Close',
'Volume', 'Dividends', 'Stock Splits', 'close_t-1', 'close_t-2', 'close_t-3',
'close_t-4', 'close_t-5', 'pct_change', 'return', 'dayofweek', 'quarter',
'month', 'year', 'dayofyear', 'dayofmonth', 'weekofyear', 'upper_band',
'middle_band', 'lower_band', 'SMA_3', 'SMA_5', 'SMA_10', 'SMA_20', 'SMA_50',
'EMA_3', 'EMA_5', 'EMA_10', 'EMA_20', 'EMA_50', 'DEMA_3', 'DEMA_5', 'DEMA_10',
'DEMA_20', 'DEMA_50', 'TEMA_3', 'TEMA_5', 'TEMA_10', 'TEMA_20', 'TEMA_50',
'ADX', 'aroon_down', 'aroon_up', 'macd', 'macdsignal', 'macdhist', 'RSI_14',
'slow_k', 'slow_d', 'williams_r', 'AD', 'OBV', 'NATR', 'TRANGE',
'fed_funds_rate', '^N225', '^IXIC', '^FTSE', '^SPX', '^DJI'], 'sequence_len':
30, 'batch_size': 16, 'permute_column': None, 'dataset_len': 5, 'horizon': 5}}}.
Seed set to 42
Fitting model to feature set: top_10:  40%|        | 2/5 [00:47<01:07,
22.54s/it]/Users/simon/anaconda3/envs/proj/lib/python3.9/site-
packages/lightning/pytorch/cli.py:520: LightningCLI's args parameter is intended
to run from within Python like if it were from the command line. To prevent
mistakes it is not recommended to provide both args and command line arguments,
got: sys.argv[1:]=['--
f=/Users/simon/Library/Jupyter/runtime/kernel-v2-6677mlgr84Kb8V46.json'],
args={'experiment_name': 'lstm_brk-b_all', 'model': {'class_path':
'src.models.LSTM_Lightning.model.StockPricePredictor', 'init_args':
{'sequence_len': 30, 'hidden_dim': 64, 'num_layers': 1, 'lr': 0.001, 'dropout':
0}}, 'data': {'class_path': 'src.models.LSTM_Lightning.data.StockDataModule',
'init_args': {'stock': 'aapl', 'feature_set': ['DEMA_3', 'Low', 'Close',
'DEMA_10', 'DEMA_20', 'EMA_5', 'Open', 'close_t-1', 'DEMA_5', 'EMA_50'],
'sequence_len': 30, 'batch_size': 16, 'permute_column': None, 'dataset_len': 5,
'horizon': 5}}}.
Seed set to 42
Fitting model to feature set: top_30:  60%|       | 3/5 [01:14<00:48,
24.38s/it]/Users/simon/anaconda3/envs/proj/lib/python3.9/site-
packages/lightning/pytorch/cli.py:520: LightningCLI's args parameter is intended
to run from within Python like if it were from the command line. To prevent
mistakes it is not recommended to provide both args and command line arguments,
got: sys.argv[1:]=['--
f=/Users/simon/Library/Jupyter/runtime/kernel-v2-6677mlgr84Kb8V46.json'],
```

```
args={'experiment_name': 'lstm_brk-b_all', 'model': {'class_path':
'src.models.LSTM_Lightning.model.StockPricePredictor', 'init_args':
{'sequence_len': 30, 'hidden_dim': 64, 'num_layers': 1, 'lr': 0.001, 'dropout':
0}}, 'data': {'class_path': 'src.models.LSTM_Lightning.data.StockDataModule',
'init_args': {'stock': 'aapl', 'feature_set': ['DEMA_3', 'Low', 'Close',
'DEMA_10', 'DEMA_20', 'EMA_5', 'Open', 'close_t-1', 'DEMA_5', 'EMA_50', 'High',
'middle_band', 'close_t-5', 'SMA_3', 'TEMA_3', 'TEMA_5', 'SMA_5', 'EMA_3',
'TEMA_20', 'TEMA_10', 'EMA_20', 'close_t-2', 'upper_band', 'SMA_20', 'SMA_10',
'DEMA_50', '^IXIC', 'EMA_10', 'close_t-4', 'TEMA_50'], 'sequence_len': 30,
'batch_size': 16, 'permute_column': None, 'dataset_len': 5, 'horizon': 5}}}.
Seed set to 42
Fitting model to feature set: top_50:  80%|      | 4/5 [01:32<00:21,
21.82s/it]/Users/simon/anaconda3/envs/proj/lib/python3.9/site-
packages/lightning/pytorch/cli.py:520: LightningCLI's args parameter is intended
to run from within Python like if it were from the command line. To prevent
mistakes it is not recommended to provide both args and command line arguments,
got: sys.argv[1:]=['--
f=/Users/simon/Library/Jupyter/runtime/kernel-v2-6677mlgr84Kb8V46.json'],
args={'experiment_name': 'lstm_brk-b_all', 'model': {'class_path':
'src.models.LSTM_Lightning.model.StockPricePredictor', 'init_args':
{'sequence_len': 30, 'hidden_dim': 64, 'num_layers': 1, 'lr': 0.001, 'dropout':
0}}, 'data': {'class_path': 'src.models.LSTM_Lightning.data.StockDataModule',
'init_args': {'stock': 'aapl', 'feature_set': ['DEMA_3', 'Low', 'Close',
'DEMA_10', 'DEMA_20', 'EMA_5', 'Open', 'close_t-1', 'DEMA_5', 'EMA_50', 'High',
'middle_band', 'close_t-5', 'SMA_3', 'TEMA_3', 'TEMA_5', 'SMA_5', 'EMA_3',
'TEMA_20', 'TEMA_10', 'EMA_20', 'close_t-2', 'upper_band', 'SMA_20', 'SMA_10',
'DEMA_50', '^IXIC', 'EMA_10', 'close_t-4', 'TEMA_50', '^SPX', 'close_t-3',
'^FTSE', 'SMA_50', '^N225', 'lower_band', 'OBV', 'dayofyear', '^DJI', 'year',
'weekofyear', 'fed_funds_rate', 'NATR', 'macd', 'RSI_14', 'month', 'ADX',
'quarter', 'macdhist', 'macdsignal'], 'sequence_len': 30, 'batch_size': 16,
'permute_column': None, 'dataset_len': 5, 'horizon': 5}}}.
Seed set to 42
Fitting model to feature set: top_50: 100%|     | 5/5 [01:49<00:00,
21.98s/it]
```

[ ]:

	feature_set	val_rmse	test_rmse
0	ohlcv	4.058506	5.781800
1	all	7.031413	15.047040
2	top_10	3.497426	4.808778
3	top_30	4.608996	7.067623
4	top_50	7.533606	15.640504

[ ]:
```
selected_fset = fset_scores.loc[fset_scores["test_rmse"].
↪idxmin()]["feature_set"]
```

[ ]:
```
feature_sets[selected_fset]
```

[ ]:
```
['DEMA_3',
 'Low',
 'Close',
 'DEMA_10',
 'DEMA_20',
 'EMA_5',
 'Open',
 'close_t-1',
 'DEMA_5',
 'EMA_50']
```

2 Rejected Approaches

2.1 Wrapper Methods

Recursive Feature Elimination

[ ]:
```
from sklearn.feature_selection import RFE
from sklearn.ensemble import RandomForestRegressor

# Select a model for feature ranking
model = RandomForestRegressor(n_estimators=100)

# RFE
selector = RFE(estimator=model, n_features_to_select=20, step=1)
selector = selector.fit(X, y)

# Selected Features
selected_features = selector.support_
selected_feature_names = X.columns[selected_features]
print("Selected features:", selected_feature_names.tolist())
```

Sequential Feature Selection (SFS)

[ ]:
```
from sklearn.feature_selection import SequentialFeatureSelector
from sklearn.linear_model import LinearRegression

# Linear model for SFS
lr = LinearRegression()

# Forward SFS (set forward=True for forward selection)
sfs = SequentialFeatureSelector(lr, n_features_to_select=10,␣
 ↪direction="forward")
sfs.fit(X, y)

# Selected Features
selected_features = sfs.get_support()
```

```
selected_feature_names = X.columns[selected_features]
print("Selected features:", selected_feature_names.tolist())
```

```
Selected features: ['Close', 'close_t-2', 'close_t-3', 'dayofweek', 'year',
'dayofmonth', 'aroon_down', 'williams_r', 'TRANGE', 'log_fed_funds_rate']
```

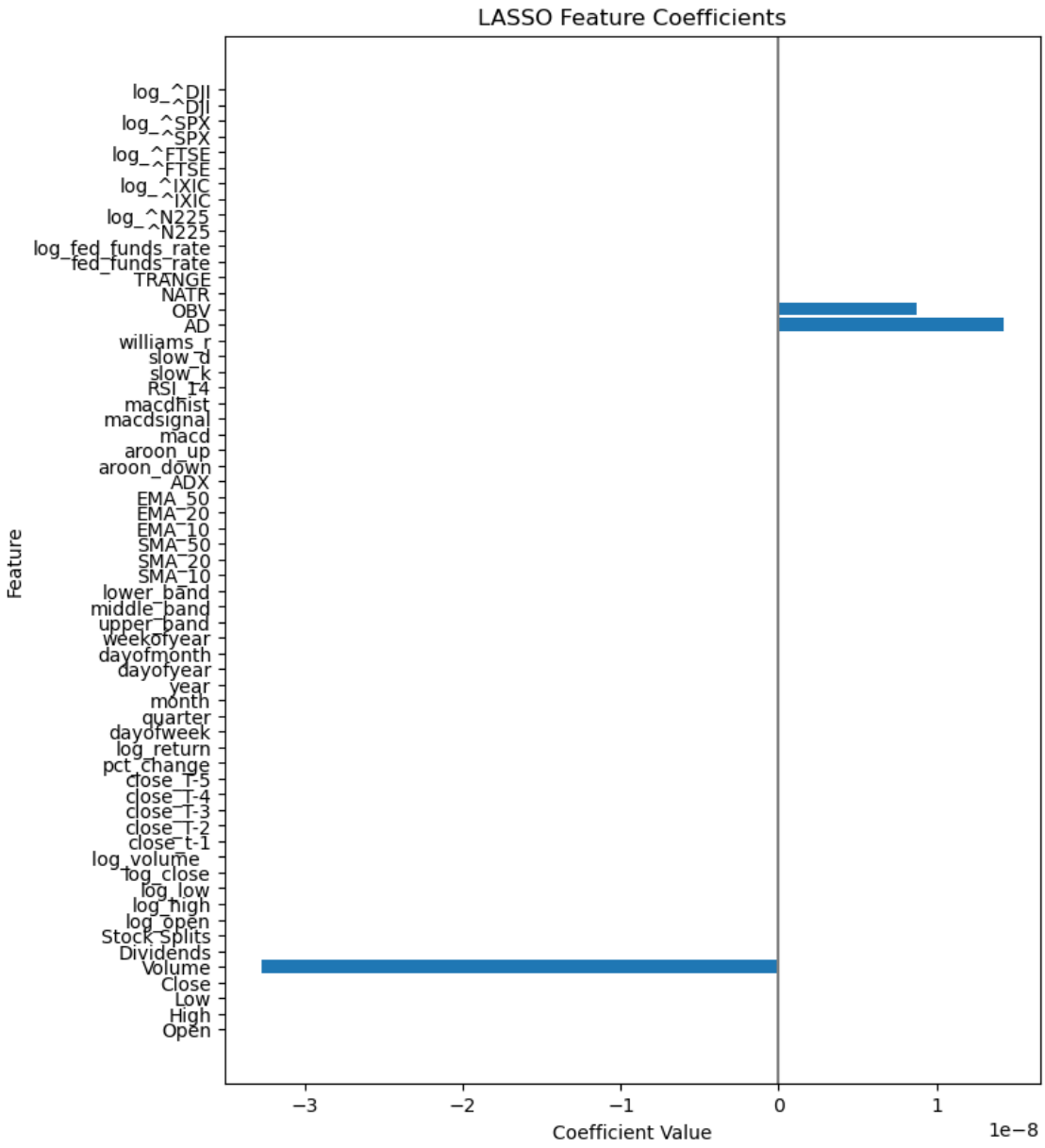
[ ]:
```
from sklearn.feature_selection import SequentialFeatureSelector
from sklearn.linear_model import LinearRegression

# Linear model for SFS
lr = LinearRegression()

# Forward SFS (set forward=True for forward selection)
sfs = SequentialFeatureSelector(lr, n_features_to_select=3,␣
 ↪direction="backward")
sfs.fit(X, y)

# Selected Features
selected_features = sfs.get_support()
selected_feature_names = X.columns[selected_features]
print("Selected features:", selected_feature_names.tolist())
```

```
Selected features: ['Close', 'log_open', 'fed_funds_rate']
```

2.2 Embedded Methods

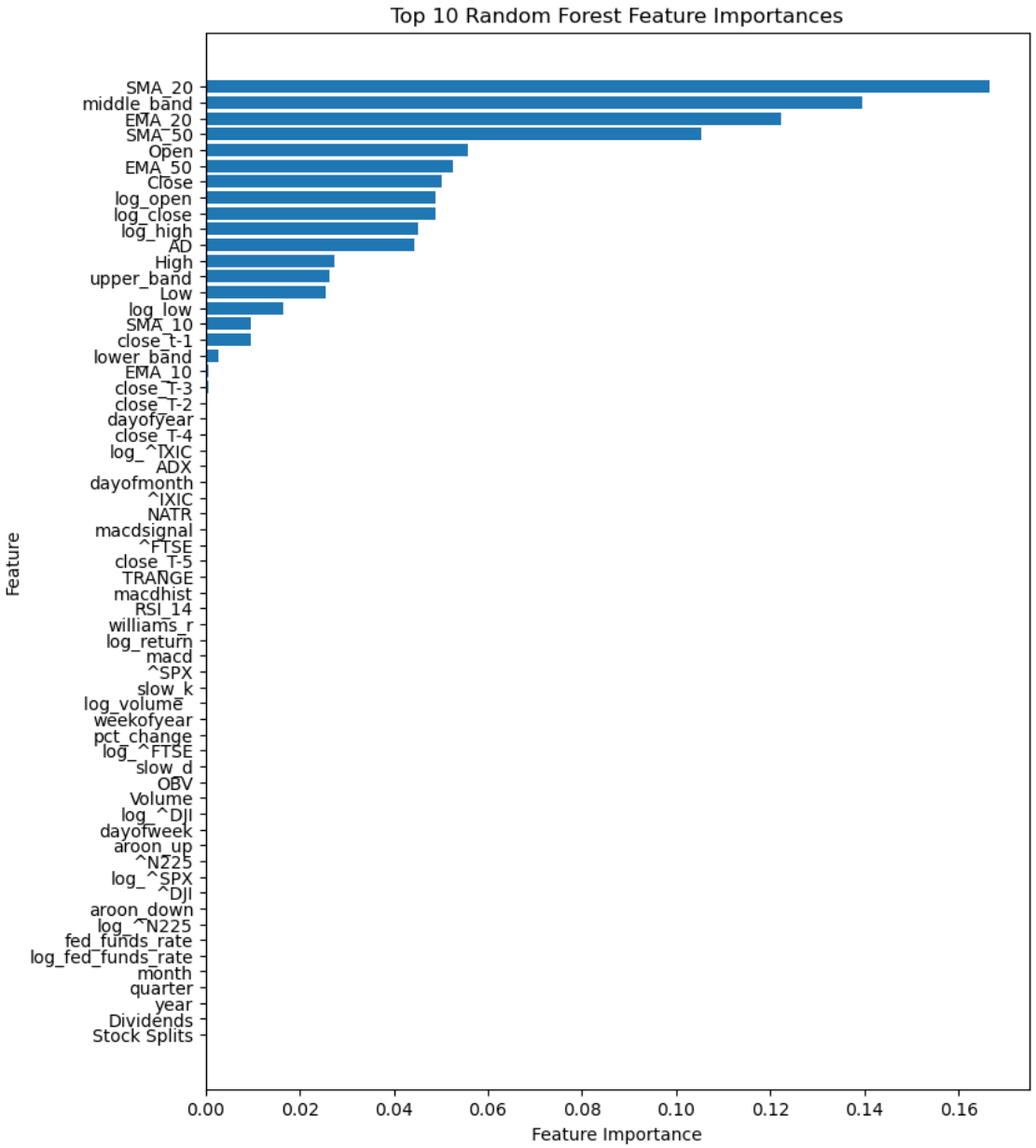
LASSO (L1 Regularization)

[ ]:
```
from sklearn.linear_model import LassoCV
import numpy as np

# LASSO with cross-validation to find the best alpha (regularization strength)
lasso = LassoCV(random_state=42).fit(X, y)

# Plot
plt.figure(figsize=(10, 10))
plt.barh(range(X.shape[1]), lasso.coef_, align="center")
plt.yticks(np.arange(X.shape[1]), X.columns)
plt.xlabel("Coefficient Value")
plt.ylabel("Feature")
plt.title("LASSO Feature Coefficients")
plt.axvline(x=0, color=".5")
plt.subplots_adjust(left=0.3)
plt.show()
```



[ ]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.datasets import make_regression

# Random Forest for feature importance
rf = RandomForestRegressor(n_estimators=100, random_state=42).fit(X, y)

# Feature importances
importances = rf.feature_importances_
indices = np.argsort(importances)[::-1]
```

```python
# Plot top N features
N = len(X.columns)  # Number of top features to display
plt.figure(figsize=(11, 11))
plt.barh(range(N), importances[indices[:N]], align="center")
plt.yticks(np.arange(N), [X.columns[i] for i in indices[:N]])
plt.xlabel("Feature Importance")
plt.ylabel("Feature")
plt.title("Top 10 Random Forest Feature Importances")
plt.subplots_adjust(left=0.3)
plt.gca().invert_yaxis()  # Invert y-axis to have the most important at the top
plt.show()
```



[ ]:
```
# Plot top N features
plt.figure(figsize=(10, 10))
plt.barh(range(N), importances[indices[:N]], align="center")
plt.yticks(np.arange(N), [X.columns[i] for i in indices[:N]])
plt.xlabel("Feature Importance")
plt.ylabel("Feature")
plt.title("Top 10 Random Forest Feature Importances")
plt.subplots_adjust(left=0.3)
plt.gca().invert_yaxis()  # Invert y-axis to have the most important at the top
plt.xlim(0, 0.0025)
plt.show()
```

2.3 Permutation Feature Importance (Column-Drop Variant)

[ ]:
```
import warnings

warnings.filterwarnings(
    "ignore", ".*Consider increasing the value of the `num_workers` argument*"
)
warnings.filterwarnings("ignore", ".*Checkpoint directory.*exists and is not␣
 ↪empty.")
warnings.filterwarnings("ignore", ".*The number of training batches (.*) is␣
 ↪smaller")

torch.manual_seed(42)

perm_importances = []
for feature_to_drop in features:
    new_feature_set = [f for f in features if f != feature_to_drop]
    model = StockPricePredictor(
        new_feature_set,
        sequence_len,
        len(new_feature_set),
        hidden_dim,
        num_layers,
        output_dim,
        lr,
    )
    data_module = StockDataModule("aapl", new_feature_set, sequence_len,␣
 ↪batch_size)
    logger = TensorBoardLogger(
        "tb_logs",
        name=f"lstm_pi_drop_{feature_to_drop}",
        default_hp_metric=False,
```

```
    )
    checkpoint_callback = ModelCheckpoint(
        dirpath="models/", save_top_k=2, monitor="hp/rmse"
    )
    trainer = Trainer(
        max_epochs=num_epochs,
        callbacks=[checkpoint_callback],
        logger=logger,
        enable_progress_bar=False,
        enable_model_summary=False,
    )
    trainer.fit(model, datamodule=data_module)

    val_metrics = trainer.validate(
        datamodule=data_module, ckpt_path="best", verbose=False
    )
    perm_importances.append(
        {
            "dropped_feature": feature_to_drop,
            "rmse": val_metrics[0]["hp/rmse"],
            "importance": val_metrics[0]["hp/rmse"] - base_rmse,
        }
    )
    print(f"{feature_to_drop}: {val_metrics[0]['hp/rmse'] - base_rmse}")
perm_importances
```

```
Open: -0.6198768615722656
High: -2.7218875885009766
Low: -1.363525390625
Close: -0.8266353607177734
Volume: -0.19973468780517578
Dividends: -0.4063596725463867
Stock Splits: 0.33273887634277344
close_t-1: -2.1524505615234375
close_t-2: -0.0051708221435546875
close_t-3: 2.136857032775879
close_t-4: -1.8818588256835938
close_t-5: -1.1553573608398438
pct_change: 1.5531063079833984
return: -1.0216188430786133
dayofweek: -0.11219978332519531
quarter: -0.026906967163085938
month: -2.866300582885742
year: -0.9714679718017578
dayofyear: -0.438201904296875
dayofmonth: -0.17662715911865234
weekofyear: -2.835394859313965
```

```
upper_band: -1.475031852722168
middle_band: -1.5739574432373047
lower_band: -1.3671636581420898
SMA_3: -1.2612180709838867
SMA_5: -0.5437421798706055
SMA_10: -1.5910024642944336
SMA_20: 1.5716371536254883
SMA_50: 0.30545711517333984
EMA_3: -2.9904356002807617
EMA_5: -0.11857223510742188
EMA_10: 0.4315004348754883
EMA_20: -1.8994951248168945
EMA_50: 1.8188304901123047
DEMA_3: -0.042777061462402344
DEMA_5: -2.2164878845214844
DEMA_10: -2.2402725219726562
DEMA_20: 1.3093843460083008
DEMA_50: 0.68011474609375
TEMA_3: -0.4061555862426758
TEMA_5: 0.6522684097290039
TEMA_10: 0.8573169708251953
TEMA_20: 1.023533821105957
TEMA_50: -1.4516239166259766
ADX: -0.9669361114501953
aroon_down: 1.5013408660888672
aroon_up: -1.4040718078613281
macd: -2.047178268432617
macdsignal: -0.3679218292236328
macdhist: -2.61270809173584
RSI_14: -0.2031717300415039
slow_k: 1.1776542663574219
slow_d: 1.3053512573242188
williams_r: 0.22715091705322266
AD: -0.2731914520263672
OBV: -1.0369043350219727
NATR: -0.2931785583496094
TRANGE: 0.2320852279663086
fed_funds_rate: 0.9436435699462891
^N225: -0.1234903335571289
^IXIC: 0.24245929718017578
^FTSE: 1.6440954208374023
^SPX: 0.36395740509033203
^DJI: 2.941483497619629
```

```
[{'dropped_feature': 'Open',
  'rmse': 13.407116889953613,
  'importance': -0.6198768615722656},
 {'dropped_feature': 'High',
```

```
 'rmse': 11.305106163024902,
 'importance': -2.7218875885009766},
{'dropped_feature': 'Low',
 'rmse': 12.663468360900879,
 'importance': -1.363525390625},
{'dropped_feature': 'Close',
 'rmse': 13.200358390808105,
 'importance': -0.8266353607177734},
{'dropped_feature': 'Volume',
 'rmse': 13.827259063720703,
 'importance': -0.19973468780517578},
{'dropped_feature': 'Dividends',
 'rmse': 13.620634078979492,
 'importance': -0.4063596725463867},
{'dropped_feature': 'Stock Splits',
 'rmse': 14.359732627868652,
 'importance': 0.33273887634277344},
{'dropped_feature': 'close_t-1',
 'rmse': 11.874543190002441,
 'importance': -2.1524505615234375},
{'dropped_feature': 'close_t-2',
 'rmse': 14.021822929382324,
 'importance': -0.0051708221435546875},
{'dropped_feature': 'close_t-3',
 'rmse': 16.163850784301758,
 'importance': 2.136857032775879},
{'dropped_feature': 'close_t-4',
 'rmse': 12.145134925842285,
 'importance': -1.8818588256835938},
{'dropped_feature': 'close_t-5',
 'rmse': 12.871636390686035,
 'importance': -1.1553573608398438},
{'dropped_feature': 'pct_change',
 'rmse': 15.580100059509277,
 'importance': 1.5531063079833984},
{'dropped_feature': 'return',
 'rmse': 13.005374908447266,
 'importance': -1.0216188430786133},
{'dropped_feature': 'dayofweek',
 'rmse': 13.914793968200684,
 'importance': -0.11219978332519531},
{'dropped_feature': 'quarter',
 'rmse': 14.000086784362793,
 'importance': -0.026906967163085938},
{'dropped_feature': 'month',
 'rmse': 11.160693168640137,
 'importance': -2.866300582885742},
{'dropped_feature': 'year',
```

```
 'rmse': 13.055525779724121,
 'importance': -0.9714679718017578},
{'dropped_feature': 'dayofyear',
 'rmse': 13.588791847229004,
 'importance': -0.438201904296875},
{'dropped_feature': 'dayofmonth',
 'rmse': 13.850366592407227,
 'importance': -0.17662715911865234},
{'dropped_feature': 'weekofyear',
 'rmse': 11.191598892211914,
 'importance': -2.835394859313965},
{'dropped_feature': 'upper_band',
 'rmse': 12.551961898803711,
 'importance': -1.475031852722168},
{'dropped_feature': 'middle_band',
 'rmse': 12.453036308288574,
 'importance': -1.5739574432373047},
{'dropped_feature': 'lower_band',
 'rmse': 12.659830093383789,
 'importance': -1.3671636581420898},
{'dropped_feature': 'SMA_3',
 'rmse': 12.765775680541992,
 'importance': -1.2612180709838867},
{'dropped_feature': 'SMA_5',
 'rmse': 13.483251571655273,
 'importance': -0.5437421798706055},
{'dropped_feature': 'SMA_10',
 'rmse': 12.435991287231445,
 'importance': -1.5910024642944336},
{'dropped_feature': 'SMA_20',
 'rmse': 15.598630905151367,
 'importance': 1.5716371536254883},
{'dropped_feature': 'SMA_50',
 'rmse': 14.332450866699219,
 'importance': 0.30545711517333984},
{'dropped_feature': 'EMA_3',
 'rmse': 11.036558151245117,
 'importance': -2.9904356002807617},
{'dropped_feature': 'EMA_5',
 'rmse': 13.908421516418457,
 'importance': -0.11857223510742188},
{'dropped_feature': 'EMA_10',
 'rmse': 14.458494186401367,
 'importance': 0.4315004348754883},
{'dropped_feature': 'EMA_20',
 'rmse': 12.127498626708984,
 'importance': -1.8994951248168945},
{'dropped_feature': 'EMA_50',
```

```
 'rmse': 15.845824241638184,
 'importance': 1.8188304901123047},
{'dropped_feature': 'DEMA_3',
 'rmse': 13.984216690063477,
 'importance': -0.042777061462402344},
{'dropped_feature': 'DEMA_5',
 'rmse': 11.810505867004395,
 'importance': -2.2164878845214844},
{'dropped_feature': 'DEMA_10',
 'rmse': 11.786721229553223,
 'importance': -2.2402725219726562},
{'dropped_feature': 'DEMA_20',
 'rmse': 15.33637809753418,
 'importance': 1.3093843460083008},
{'dropped_feature': 'DEMA_50',
 'rmse': 14.707108497619629,
 'importance': 0.68011474609375},
{'dropped_feature': 'TEMA_3',
 'rmse': 13.620838165283203,
 'importance': -0.4061555862426758},
{'dropped_feature': 'TEMA_5',
 'rmse': 14.679262161254883,
 'importance': 0.6522684097290039},
{'dropped_feature': 'TEMA_10',
 'rmse': 14.884310722351074,
 'importance': 0.8573169708251953},
{'dropped_feature': 'TEMA_20',
 'rmse': 15.050527572631836,
 'importance': 1.023533821105957},
{'dropped_feature': 'TEMA_50',
 'rmse': 12.575369834899902,
 'importance': -1.4516239166259766},
{'dropped_feature': 'ADX',
 'rmse': 13.060057640075684,
 'importance': -0.9669361114501953},
{'dropped_feature': 'aroon_down',
 'rmse': 15.528334617614746,
 'importance': 1.5013408660888672},
{'dropped_feature': 'aroon_up',
 'rmse': 12.62292194366455,
 'importance': -1.4040718078613281},
{'dropped_feature': 'macd',
 'rmse': 11.979815483093262,
 'importance': -2.047178268432617},
{'dropped_feature': 'macdsignal',
 'rmse': 13.659071922302246,
 'importance': -0.3679218292236328},
{'dropped_feature': 'macdhist',
```

```
 'rmse': 11.414285659790039,
 'importance': -2.61270809173584},
{'dropped_feature': 'RSI_14',
 'rmse': 13.823822021484375,
 'importance': -0.2031717300415039},
{'dropped_feature': 'slow_k',
 'rmse': 15.2046480178833,
 'importance': 1.1776542663574219},
{'dropped_feature': 'slow_d',
 'rmse': 15.332345008850098,
 'importance': 1.3053512573242188},
{'dropped_feature': 'williams_r',
 'rmse': 14.254144668579102,
 'importance': 0.22715091705322266},
{'dropped_feature': 'AD',
 'rmse': 13.753802299499512,
 'importance': -0.2731914520263672},
{'dropped_feature': 'OBV',
 'rmse': 12.990089416503906,
 'importance': -1.0369043350219727},
{'dropped_feature': 'NATR',
 'rmse': 13.73381519317627,
 'importance': -0.2931785583496094},
{'dropped_feature': 'TRANGE',
 'rmse': 14.259078979492188,
 'importance': 0.2320852279663086},
{'dropped_feature': 'fed_funds_rate',
 'rmse': 14.970637321472168,
 'importance': 0.9436435699462891},
{'dropped_feature': '^N225',
 'rmse': 13.90350341796875,
 'importance': -0.1234903335571289},
{'dropped_feature': '^IXIC',
 'rmse': 14.269453048706055,
 'importance': 0.24245929718017578},
{'dropped_feature': '^FTSE',
 'rmse': 15.671089172363281,
 'importance': 1.6440954208374023},
{'dropped_feature': '^SPX',
 'rmse': 14.390951156616211,
 'importance': 0.36395740509033203},
{'dropped_feature': '^DJI',
 'rmse': 16.968477249145508,
 'importance': 2.941483497619629}]
```

[ ]:
```
df = (
    pd.DataFrame(perm_importances)
```

```
    .sort_values(by="importance", ascending=False)
    .reset_index()
)
# Plotting
plt.figure(figsize=(12, 10))
plt.barh(df["dropped_feature"], df["importance"], color="skyblue")
plt.xlabel("Importance")
plt.ylabel("Dropped Feature")
plt.title("Permutation Feature Importance (Column-Drop Variant)")
plt.tight_layout()
plt.show()
```