NUMERAI

Independent Research: Data Pipeline + Model Dev for NumerAI Hedge Fund(rough d)

By: Brandonlee Santos

Date: 02-20-23

Data Pipeline

NumerAI EC2 instance containing the current encrypted data

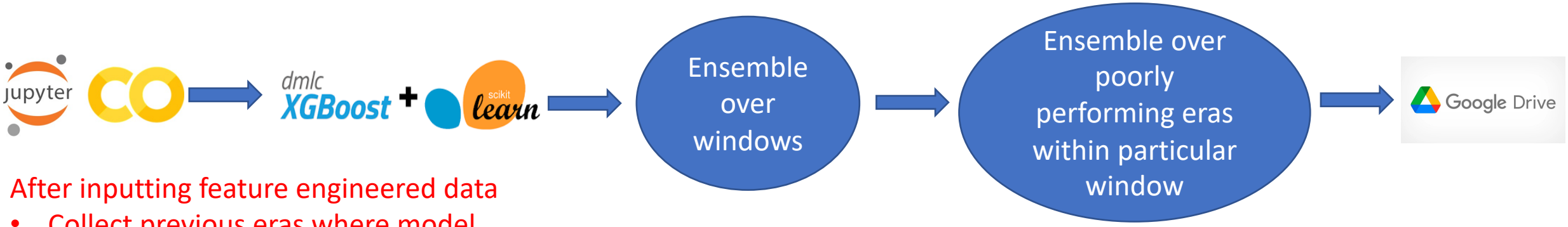
NumerAI API feeds into CoLab virtual machine constraints:(125GB HDD, 25GB RAM)

Python Script reads in from Google Drive and performs following tasks:

- Numerical col np.array dtype to float16 in order to preserve RAM
- Era col to int64
- Slice training data in half(for memory constraints) and into windows for future ensemble (reducing variance)

- Final feature engineering step is to LabelEncode the Era col.
- Lastly, tree based feature selection works well by removing bottom ~ 5% (nonscientific recommended range 0-15%) from Mean Decrease Accuracy technique from a XGBoost Regressor, which was hyperparameter tuned via GridSearchCV using Purged Window Time-Series cross validation
- The grid for the GridSearchCV should contain few hyperparameters as temporal constraints limit us
- Save pandas Dataframe manipulations to Google drive

Model Pipeline



After inputting feature engineered data

- Collect previous eras where model preforms poorly and ensemble over
- Consider example grid on the right; optimize as many hyperparameter as time permits
- Ensemble over windows
- Ensemble over different models works very well
- Pickle save model .pkl
- BayesSearchCV, hyperopt seem to be more promising but take up more time

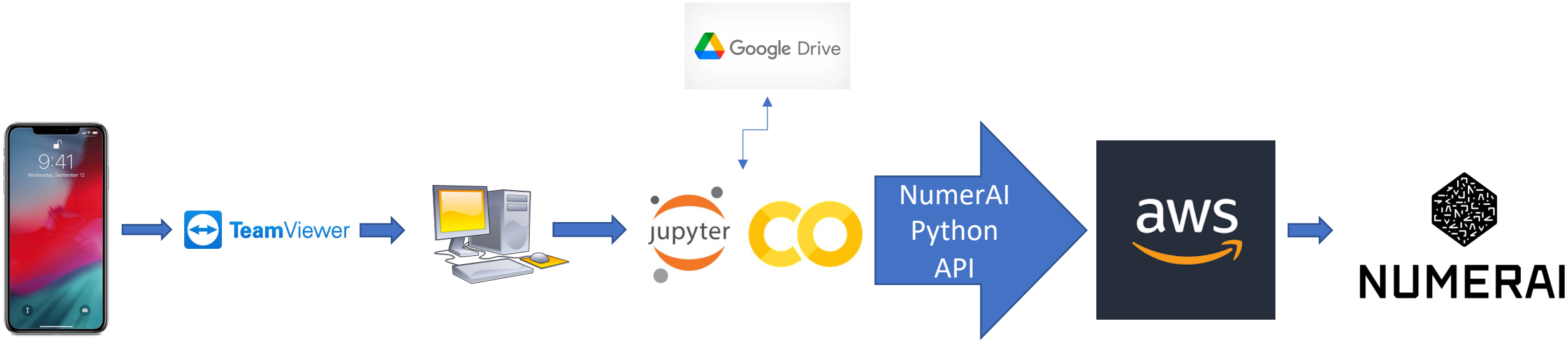
```
gridg2  = {
    'learning_rate': [0.03, 0.01, 0.03, 0.009],
    'bagging_temperature': [0, 1,5,10],
     'n_estimators':np.arange(100,250,50),
     'reg_alpha': [0.1, 0.5],
     'reg_lambda':[0, 1, 1.5],
     'num_leaves':np.arange(25,40,5),
        'alpha':[0,1,1.5],
        'max_depth':[6,9,12],
        'min_child_weight':[1,2,3],
        'gamma':np.arange(0,1.1,0.1),
        'colsample_bytree':[0.9,1]
     }
```

CV method: Purged Time- Series Window Split

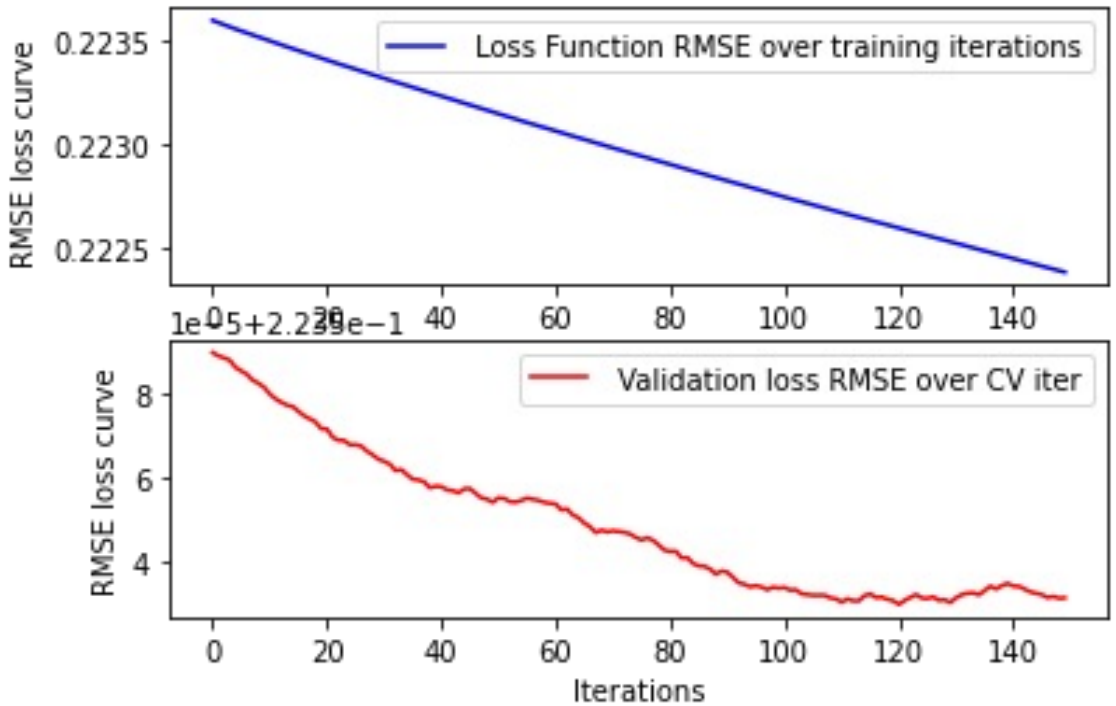
```python
class PurgedTimeSeriesSplitGroups(_BaseKFold):
    def __init__(self,groups, n_splits=5, purge_groups=0):
        super().__init__(n_splits, shuffle=False, random_state=None)
        self.purge_groups = purge_groups
        self.groups = groups

    def split(self, X, y=None, groups=None):
        X, y, groups = indexable(X, y, groups)
        groups = self.groups
        n_samples = _num_samples(X)
        n_folds = self.n_splits + 1
        group_list = np.unique(groups)
        n_groups = len(group_list)
        if n_folds + self.purge_groups > n_groups:
            raise ValueError((f"Cannot have number of folds plus purged groups "
                              f"={n_folds+self.purge_groups} greater than the "
                              f"number of groups: {n_groups}."))
        indices = np.arange(n_samples)
        test_size = ((n_groups-self.purge_groups) // n_folds)
        test_starts = [n_groups-test_size*c for c in range(n_folds-1, 0, -1)]
        for test_start in test_starts:
            yield (indices[groups.isin(group_list[:test_start-self.purge_groups])],
                   indices[groups.isin(group_list[test_start:test_start + test_size])])
```

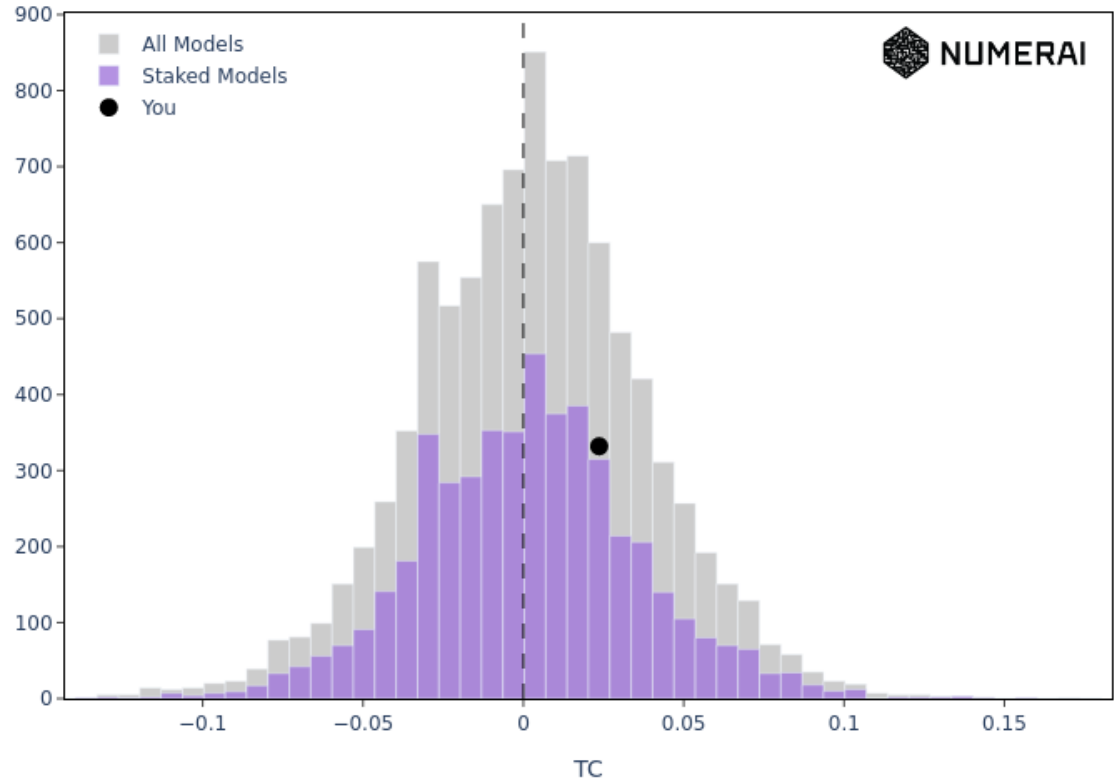
- Override the groups by inputting the era column.
- This is the trivial Time-Series window split, however respects the era boundaries to provide consistent forecasting

Semi-Automated Submission Pipeline



Results


RMSE loss curve
0.2235
0.2230
0.2225
Loss Function RMSE over training iterations
0
20
40
60
80
100
120
140
1e−5+2.239e−1
RMSE loss curve
8
6
4
Validation loss RMSE over CV iter
0
20
40
60
80
100
120
140
Iterations


900
800
700
600
500
400
300
200
100
0
All Models
Staked Models
You
NUMERAI
−0.1
−0.05
0
0.05
0.1
0.15
TC

Some conclusions and ways to improve

- Topological methods such as UMAP, maybe even persistent homology, potentially differential geometric regression methods.
- Consider using other targets as training although this ruined my model proceed with caution
- Ensemble of models is the general consensus of the best direction to head in
- Creating windows of eras and ensemble over is a fantastic approach to reduce variance and exploit ensemble learning
- Ensemble over the poor performing eras and retrain a model over those
- BayesianSearchCV is much more promising than GridSearchCV however is not feasible due to time-complexity
- Feature neutralization is a great technique to reduce the nonstationary of the prediction however my VM is limited with 25Gb and cannot solve the SVD problem
- The more data and the more ensemble generally the better the results
- PCA, LDA and other transformation do not seem to work well due to the high dimensional nature of the data and the linearity imposed in those methods
- AutoEncoder feature synthesis is an interesting approach that people claim in the forum and RocketChat produces great models
- Outlier removal seems to also be prevalent online although tree based models handle this built in

Resources

- https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3478927
- NumerAI fireside chats