Wind Power Scenario Generation

With Machine Learning Algorithms

University of Washington
Daehyun Kim

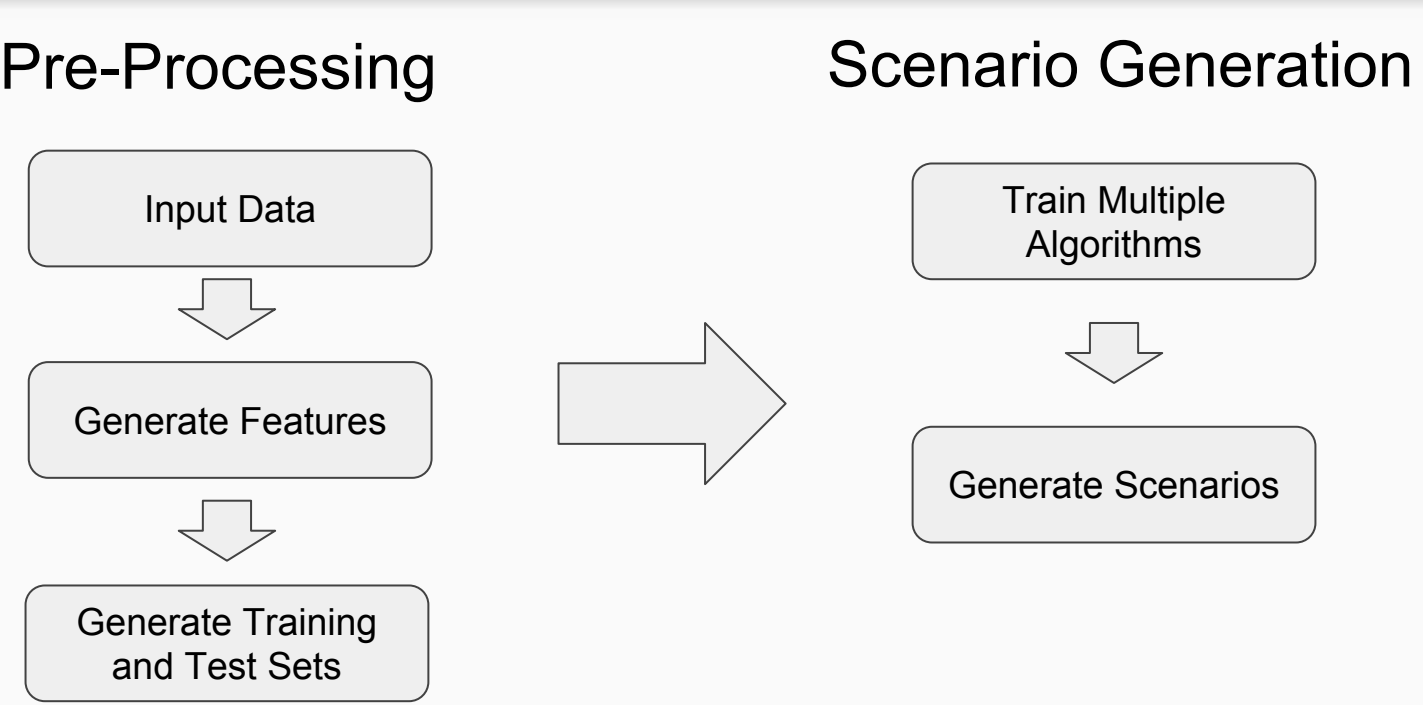
Background

- Meeting the electricity demand in a reliable way
- Wind energy is one of the key sustainable energy sources
- Balance between reliability and economics
- Uncertainty and variability
- Multiple prediction algorithms developed by researchers
- Scenario generation for stochastic optimization

Algorithms to Explore

- Neural Networks
  - Multi-Layer Perceptron - Input, Hidden, and Output Layers.
- Random Forest
  - Ensembling, Bootstrapping, Combine Decision Trees with Random Subsets with averages
- Gradient Boosted tree
  - Ensembling, Boosting, Add to already trained ensemble, Focus on weak learners with weighted averages
- Nearest Neighbor
  - Simple machine learning, Average of the K nearest data points
- Linear Regression
- Kernel Ridge Regression
  - Regularization, Ridge loss
- Support Vector Machines
  - Decision Plane that maximizes the margin, Subset of training data, epsilon-insensitive loss

Flowchart


Pre-Processing
Input Data
Generate Features
Generate Training and Test Sets
Scenario Generation
Train Multiple Algorithms
Generate Scenarios

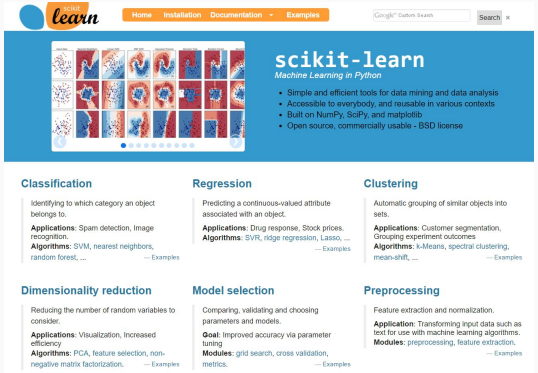
Pre-Processing

- NREL Western Dataset in 2006
  - Wind Power and Wind Speed
  - Power normalized based on capacity
- Generate Features
  - Time Leads - hours ahead prediction
  - Local Historical Hours
  - Resolution
- Generate Training, Testing Sets
  - Training Length

```
    for iLocation in range(nLocation):
        wf_id = wf_idx[iLocation] # the name (number) of iLocation in RTS
        farm_idx = wf_name[iLocation] # pick wind sites in ith Location
        nSite = len(farm_idx) # number of sites in iLocation
        turbine_param = np.zeros((nSite, nRow, 4)) # parameters in each farm

        for iSite in range(nSite):
            with open('./2006/2006/' + str(wf_id) + '/' + str(farm_idx[iSite]) + '.csv') as f:
                reader = csv.reader(f)
                next(reader)
                count = 0
                for row in reader:
                    turbine_param[iSite, count, :] = row[1:]
                    count += 1
        loc_capacity = 30*nSite
        capacity.append(loc_capacity)
        wind_param.append(turbine_param)
        speed_temp.append(np.mean(turbine_param[:,:,0],axis=0))
        gen_temp.append(np.sum(turbine_param[:,:,3], axis=0)/(loc_capacity))

        if resolution == 1:
            speed_per_hour = np.reshape(speed_temp[iLocation], (nRow//6, 6))
            gen_per_hour = np.reshape(gen_temp[iLocation], (nRow//6, 6))
            speed.append(np.mean(speed_per_hour, axis=1)/30)
            gen.append(np.mean(gen_per_hour, axis=1))
```

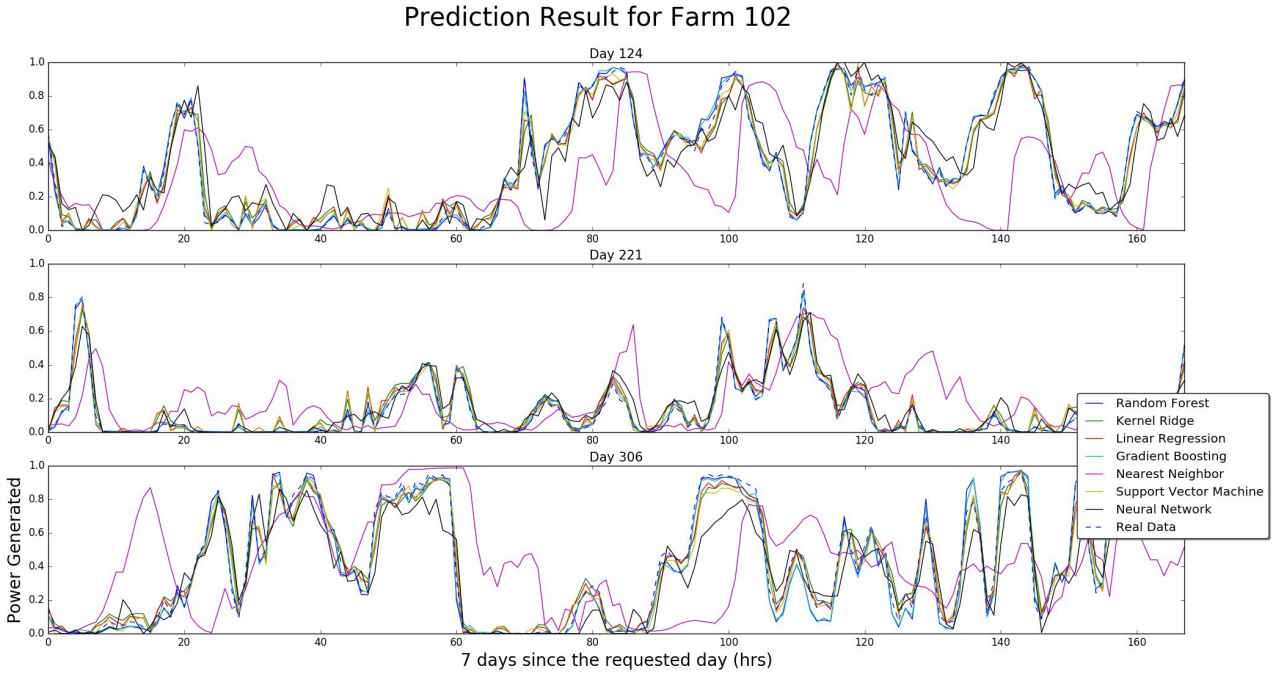
Scenario Generation

- Tool Package (open source)
  - Scikit-Learn
- Multiple Algorithms

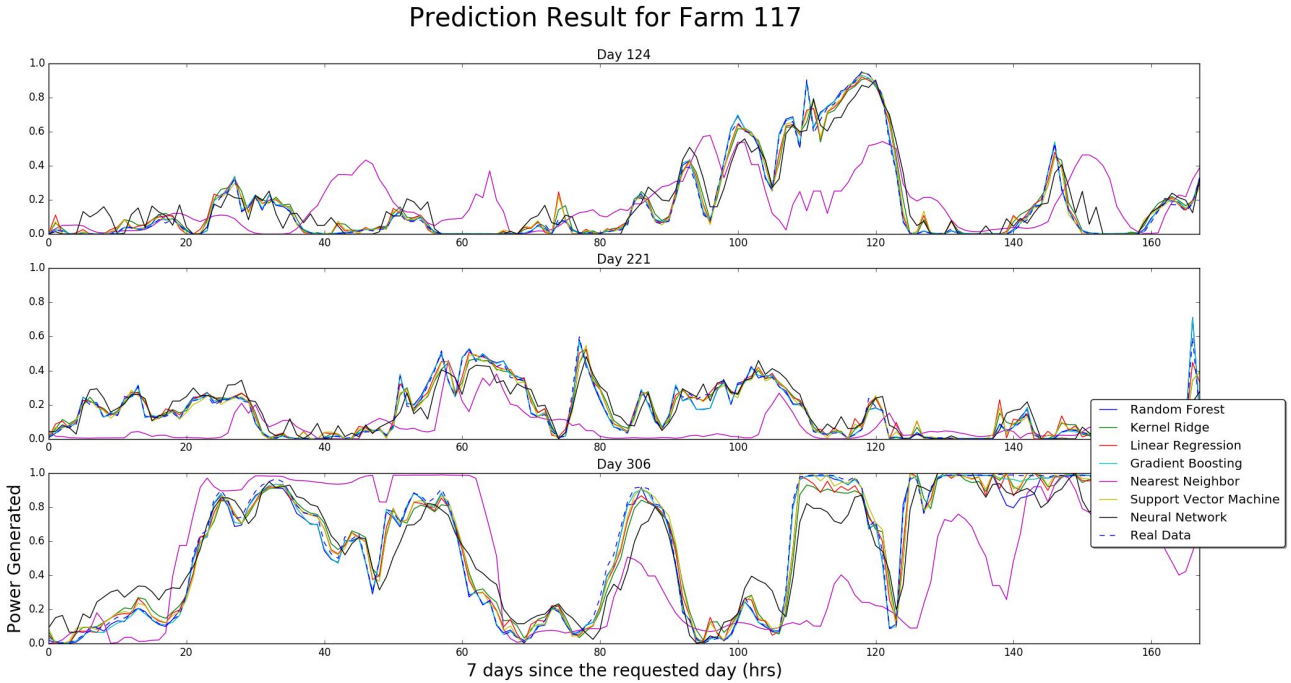



```
    Estimators = {
                "Linear Regression": linear_model.LinearRegression(),
                "Support Vector Machine": svm.LinearSVR(),
                "Kernel Ridge": kernel_ridge.KernelRidge(),
                "Random Forest": RandomForestRegressor(),
                "Gradient Boosting": GradientBoostingRegressor(),
                "Neural Network": neural_network.MLPRegressor(),
                "Nearest Neighbor": neighbors.KNeighborsRegressor(),
    }

    y_test_predict = dict()

    for name, estimator in Estimators.items():
        t1 = time.time()
        print (name, "------")
        estimator.fit(xTr[f], yTr[f].ravel())
        y_test_predict[name] = estimator.predict(xTe[f])
        rmse = math.sqrt(np.mean((y_test_predict[name] - yTe[f])**2))
        mae = np.mean(abs(y_test_predict[name] - yTe[f]))
        t2 = time.time()
        print ("Coefficient of Determination:", estimator.score(xTe[f], yTe[f]))
        print ("Root-Mean-Squared Error:", rmse)
        print ("Mean Absolute Error:", mae)
        print ("Time for each algorithm:", t2-t1)
```

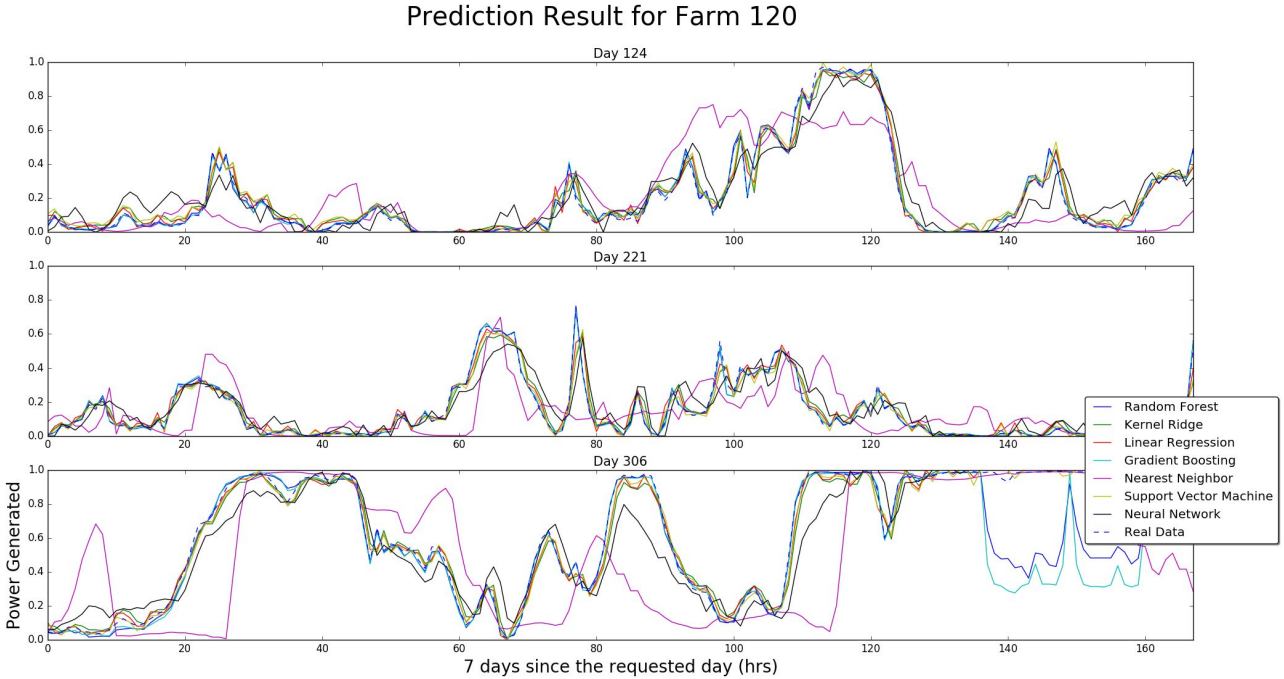
Results - Farm 1



Results - Farm 4



Results - Farm 7

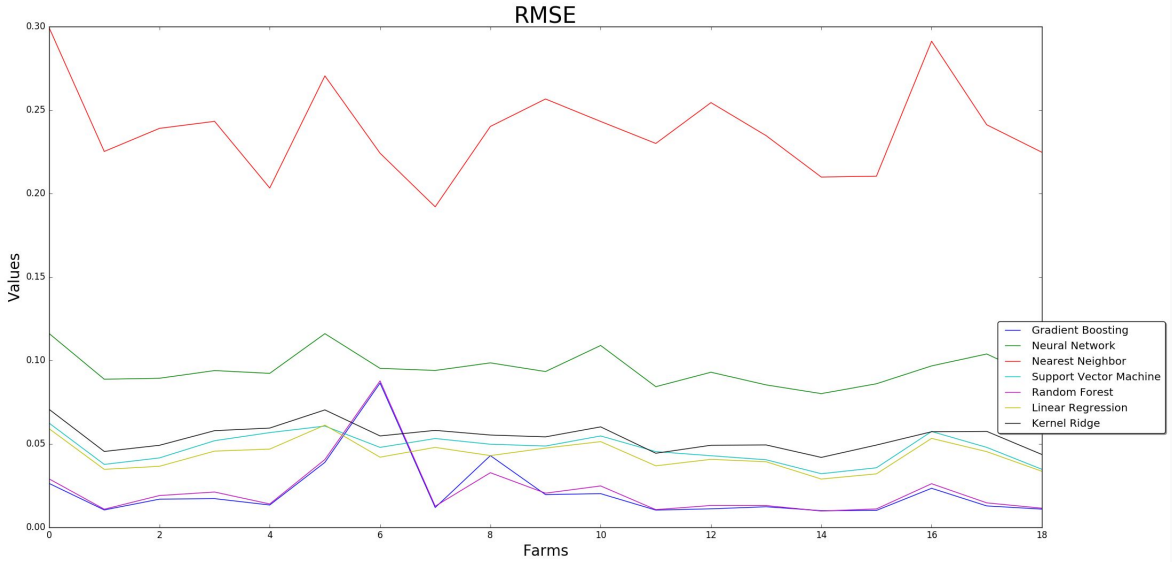


Results - RMSE

- Averaged on day

  124, 221, 306

- Best:
  - Random Forest
  - Gradient Boosting
- Worst:
  - Nearest Neighbor
  - Neural Networks

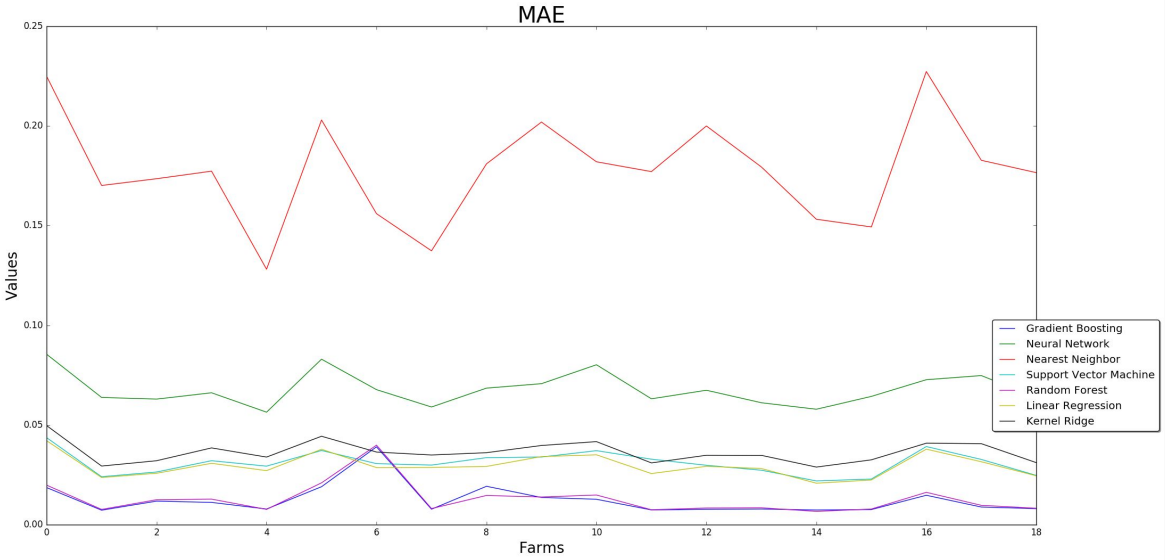


Results - MAE

- Averaged on day

  124, 221, 306

- Best:
  - Random Forest
  - Gradient Boosting
- Worst:
  - Nearest Neighbor
  - Neural Networks

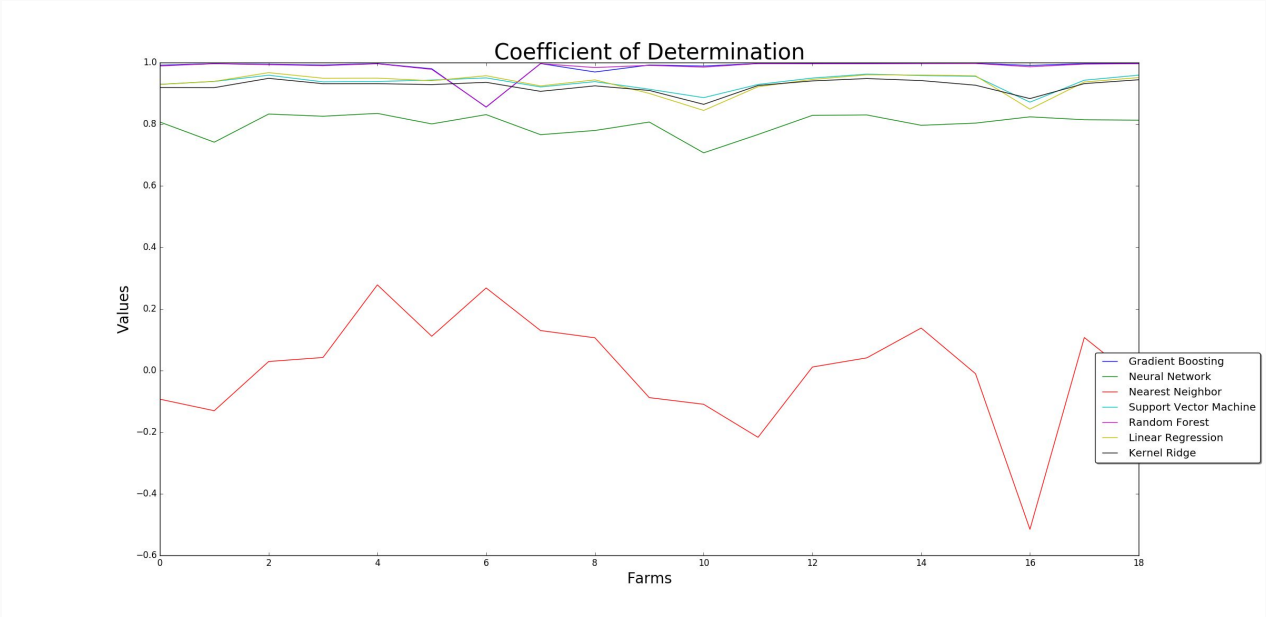


Results - Coefficient of Determination

- Averaged on day

  124, 221, 306

- Best:
  - Random Forest
  - Gradient Boosting
- Worst:
  - Nearest Neighbor
  - Neural Networks



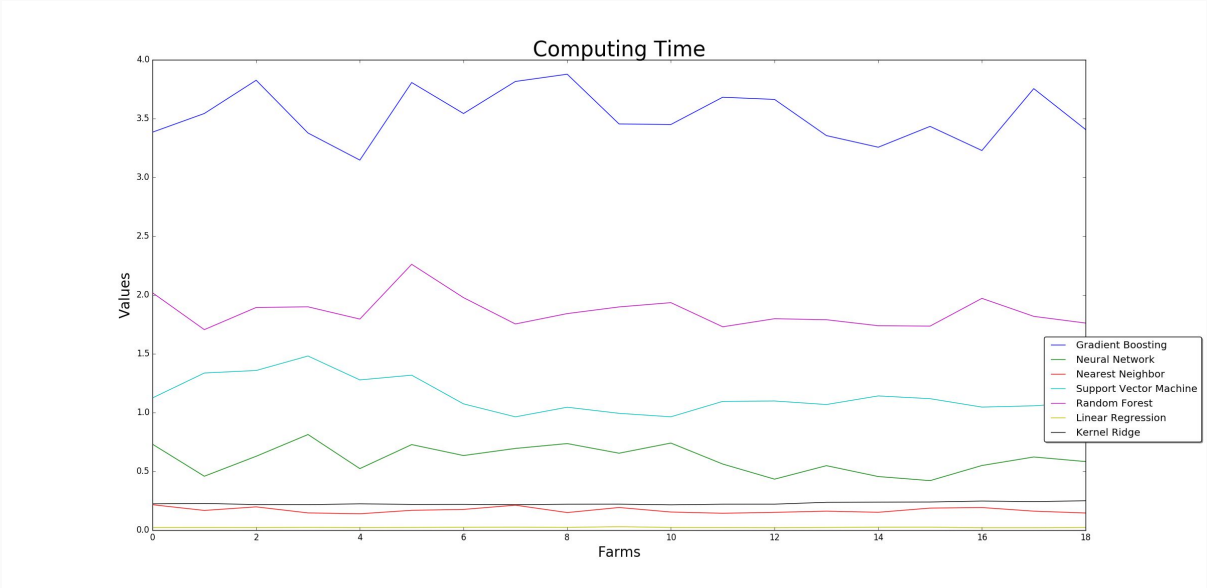
Results - Computing Time

- Averaged on day

  124, 221, 306

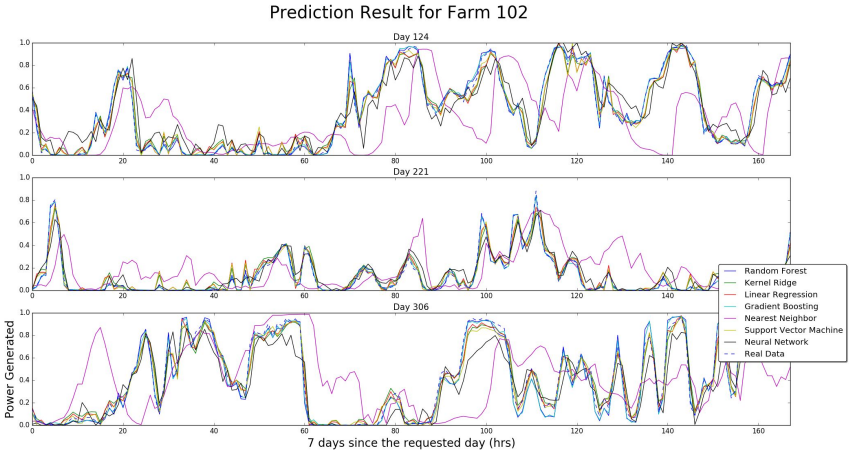
- Best:
  - Linear Regression
  - Kernel Ridge
  - Nearest Neighbor
- Worst:
  - Random Forest
  - Gradient Boosting

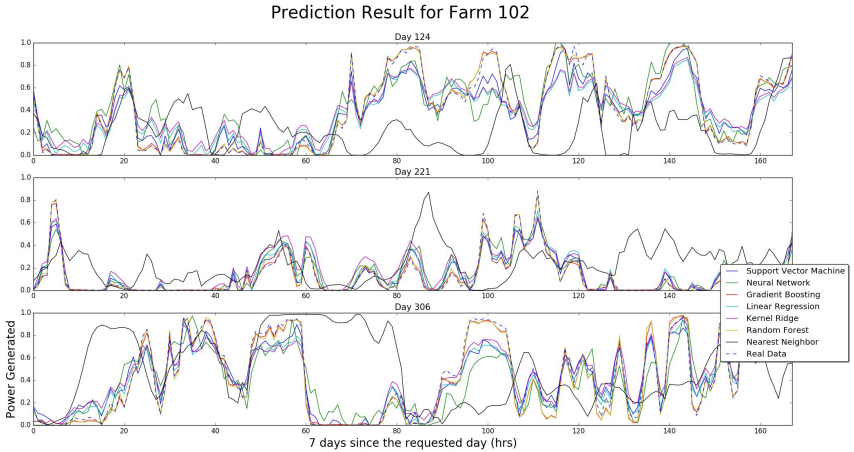
Script: Python
OS: Windows 10
SW: Eclipse, Python 3.5.1
Machine: ASUS Q550L
Intel Core i7



Variations - Time Leads

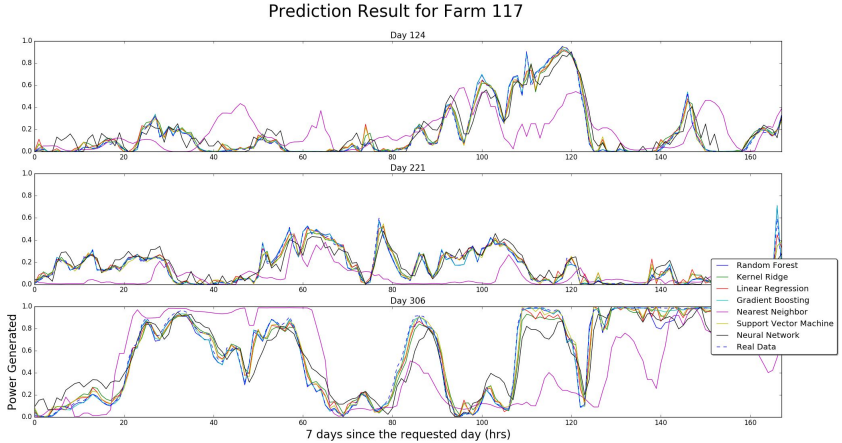
Left -> time_lead = 1, Right -> time_lead = 24

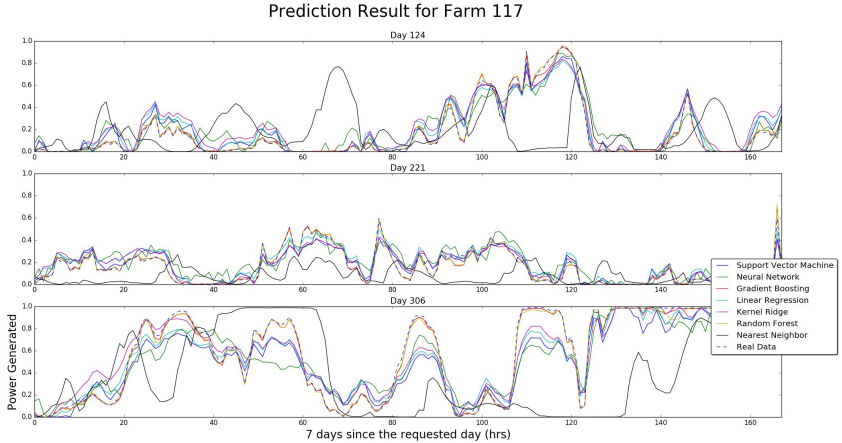




Variations - Time Leads

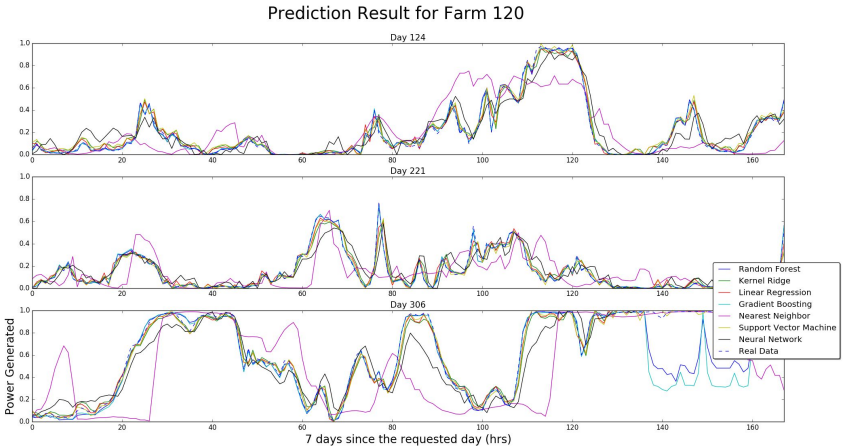
Left -> time_lead = 1, Right -> time_lead = 24

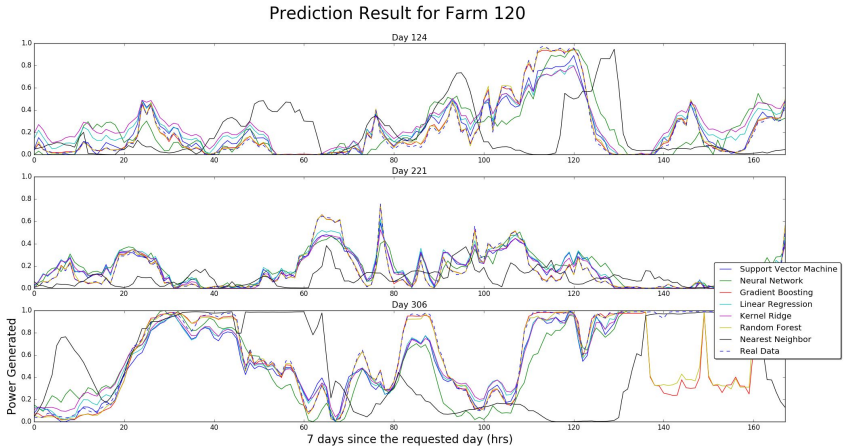




Variations - Time Leads

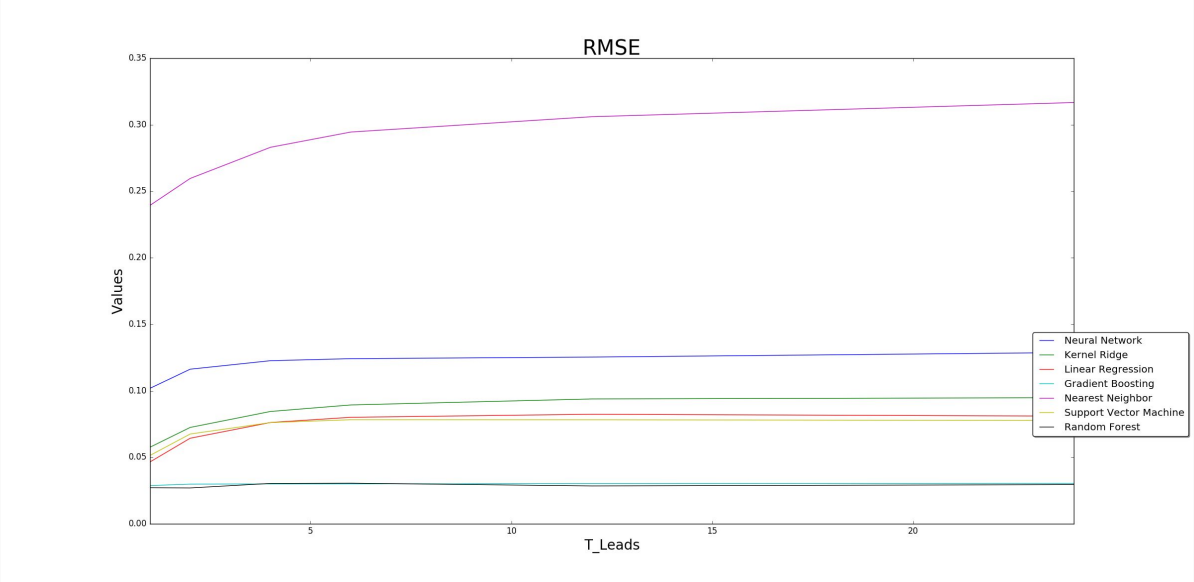
Left -> time_lead = 1, Right -> time_lead = 24





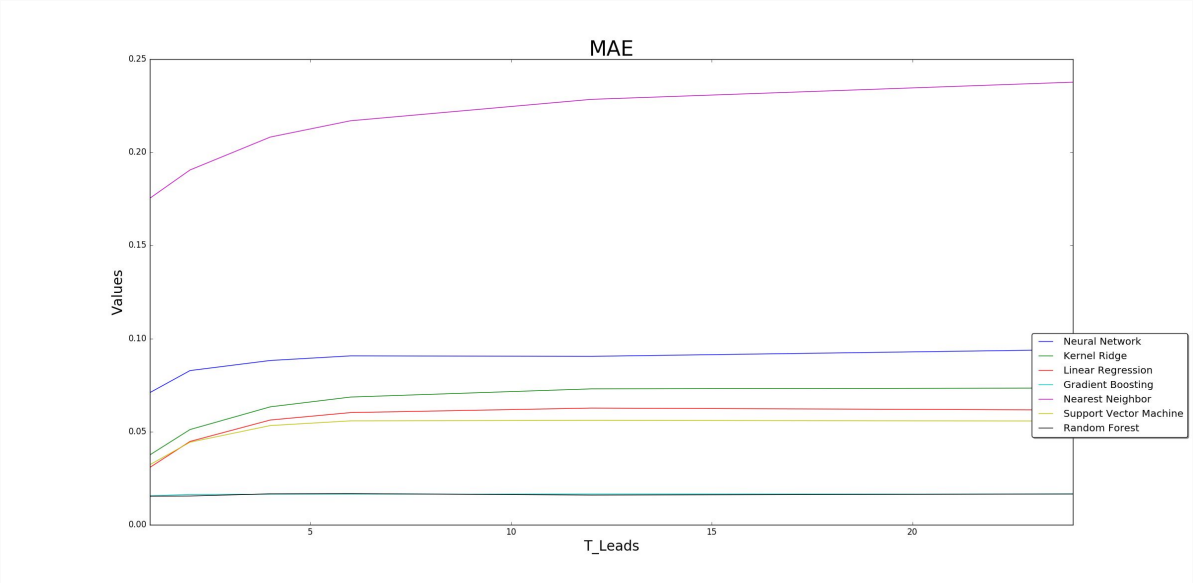
Variations - Time Leads

- Tested on
  - t_lead = 1, 2, 4, 6, 12, 24
- Averaged on day
  - 124, 221, 306
  - & first 10 farms
- Errors rise



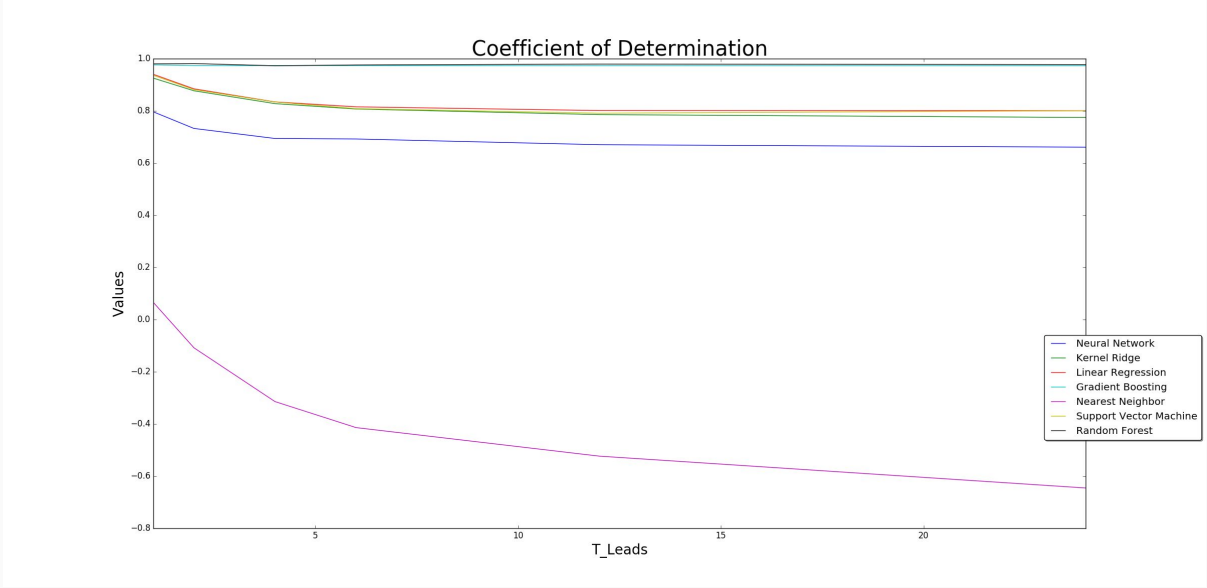
Variations - Time Leads

- Tested on
  - t_lead = 1, 2, 4, 6, 12, 24
- Averaged on day
  - 124, 221, 306
  - & first 10 farms
- Errors rise



Variations - Time Leads

- Tested on
  - t_lead = 1, 2, 4, 6, 12, 24
- Averaged on day
  - 124, 221, 306
  - & first 10 farms
- Goodness of Fit decreases



Future Studies

- Spatial features
- Cross validation
- Sampling multiple scenarios from each algorithm
- Testing different open-source packages
- Testing different data

Questions?