CS 7641 Assignment 1: Supervised Learning

Introduction

This writeup contains answers to CS7641 Assignment 1: Supervised Learning.

It compares 5 Supervised Machine Learning classification algorithms:

- **Decision trees**
- **Boosting**
- **Support vector machines (2 kernels: poly and rbf)**
- **Neural networks**
- **K nearest neighbors**

On 2 Datasets:

- **Wind speed:** https://www.openml.org/d/503 and https://www.openml.org/d/847
- **Customer retention:** https://www.openml.org/d/42178

For each algorithm on both datasets following results are analyzed:

- **Model complexity curves:**

  A model with given number of training instances is trained with different values of hyperparameters and training & cross-validation (CV) scores are plotted against different values of hyperparameters such that complexity of model increases left to right.

  Typically, as model starts becoming more complex both training score & CV scores improve. But later, as model becomes overly complex CV score decreases while training score keeps on improving, which is an indicator of overfitting. Model complexity curves helps us fine tune the optimal values of hyperparameters such that the trained model is complex enough but not too complex.

- **Learning curves:**

  A model with tuned values of hyperparameters is trained with increasing number of training instances and resulting training and cross-validation (CV) scores and times are plotted as function of number of training instances.

  Typically, these curves rise sharply initially and then flattens out. These curves help us understand if and how number of training instances are helping in improving scores and/or times. Which in turn helps us decide if getting new instances for training will

improve the model or other actions like retuning the hyperparameters or dimensionality reduction needs to be performed.

- **Out-of-sample model evaluation metrics:**

  The CV scores are not true estimates of the model performance on unseen data, as CV set itself was used to fine tune hyperparameter values. Typically, CV scores are overestimates of the performance on the unseen data. To get a true estimate of model performance on unseen data, a small set (~20%) of training data is held-out at the beginning and the model performance is reported on this set.

These results are the compared based on:

- **Cross validation scores:**

  CV score for each of the tuned and trained model is plotted as a function of number of training instances. Typically, as number of training instances increases CV scores increases as well. These curves help us compare the rate of increase of CV scores with number of training instances.

- **Training times:**

  Training times for each of the tuned and trained model is plotted as a function of number of training instances. Typically, training times also grows with number of training instances. These curves help us compare the rate of increase in training times with number of training instances.

- **Model evaluation metrics:**

  Model evaluation metrics for each tuned and trained model is compared. This gives a true estimate of model performance on unseen data.

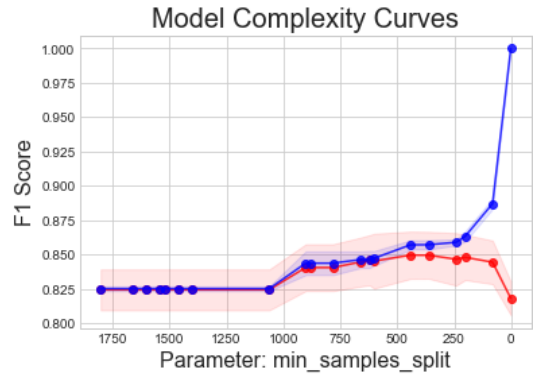
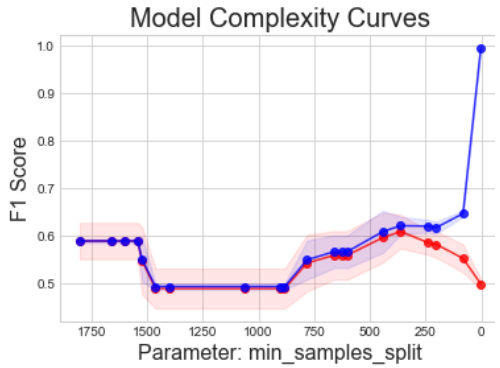
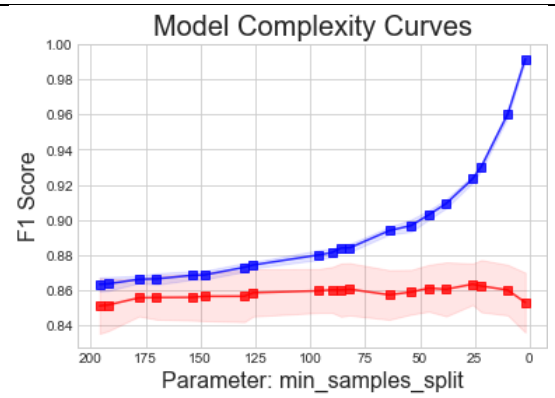
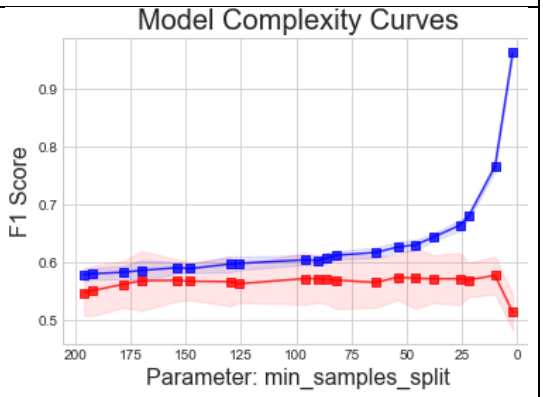
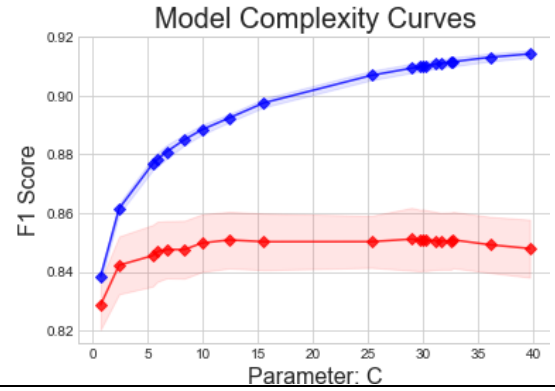
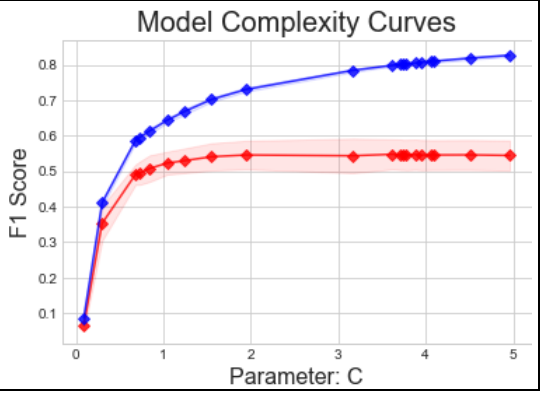
Lastly, comprehensive conclusions and future improvements are discussed.

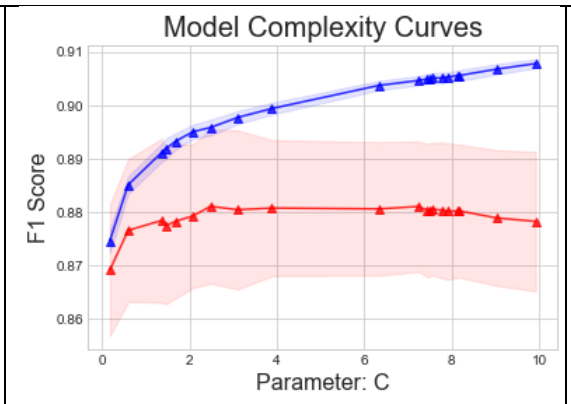
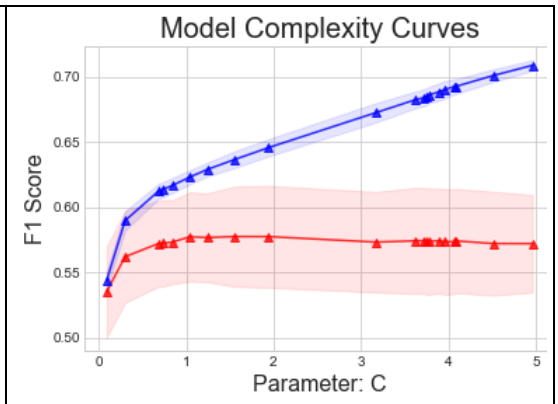
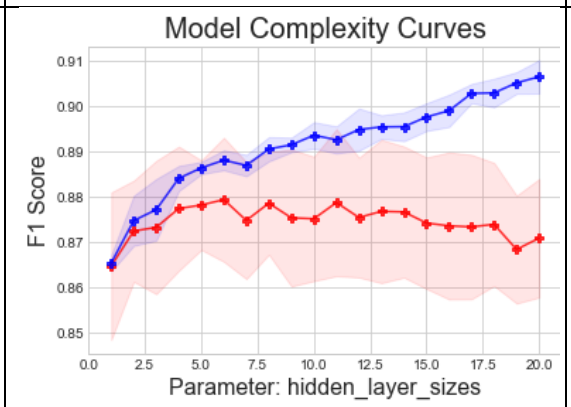
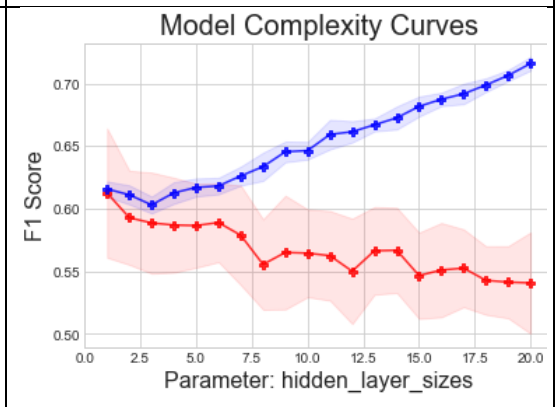
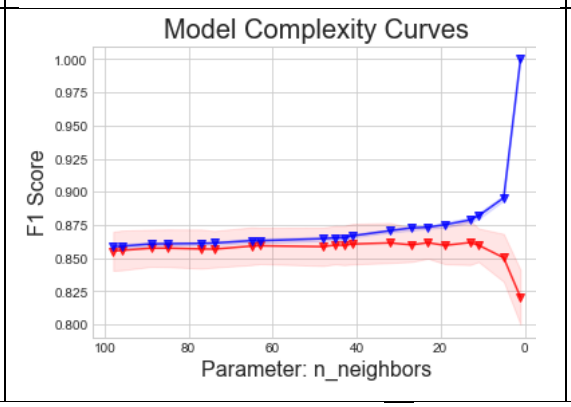
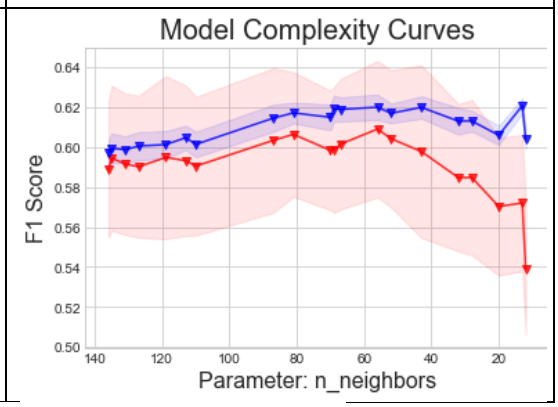
Implementation

- Most of the code is abstracted out in helpers.py and imported in iPython notebook.
- iPython notebook contains all the code along with analysis.
- Following 3 steps are performed on each dataset mentioned above:
    - **Preprocessing:**
        - Data is loaded from csv files downloaded from sources mentioned above.
        - Null values (if any) are handled depending on the use case.
        - LabelEncoding is performed on target variable.
        - OneHotEncoding is performed on categorically variables (if any).
        - StandardScaling is performed on all variables.
        - Data is split in X and y.
        - Data is split in train and test sets.
    - **Machine learning:**
        - 6 models mentioned above are trained on both the dataset mentioned above.
        - RandomSearchCV is used to fit hyperparameters.
        - 10-fold cross-validation (CV) is performed during RandomSearchCV.
        - Model complexity curves are plotted.
        - Learning curves are plotted for the "best" model (F1 score is used to define best).
        - Finally, model evaluation metrics are calculated and printed for out-of-sample held-out set.
    - **Model comparison:**
        - After all 6 machine learning model are successfully trained and tuned, CV scores, training times and model evaluation metrics are compared.

Results

Model complexity curves

Algorithm	Dataset#1: Wind speed	Dataset#2: Customer retention
Decision Tree		
Boosting		
SVM (Kernel: poly)		

SVM (Kernel: rbf)		
Neural Network		
K Nearest Neighbors		

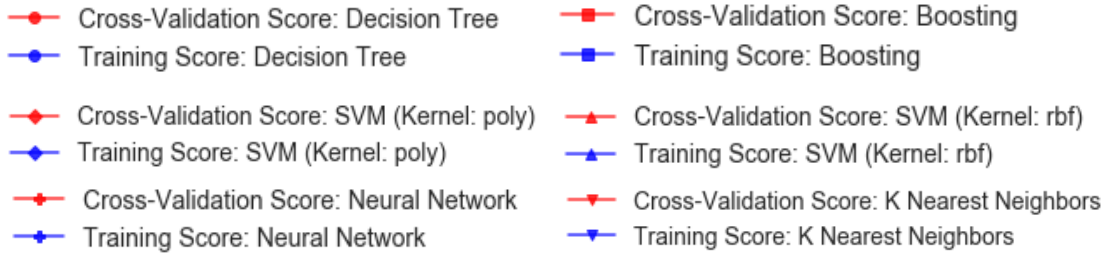
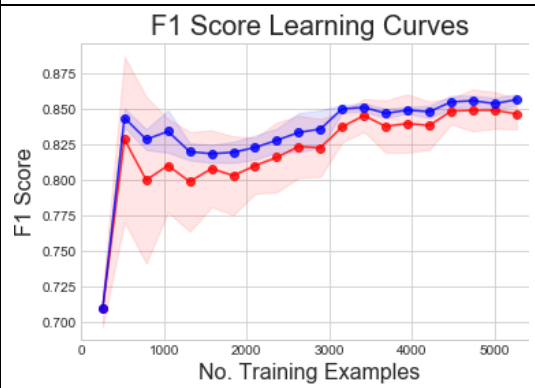
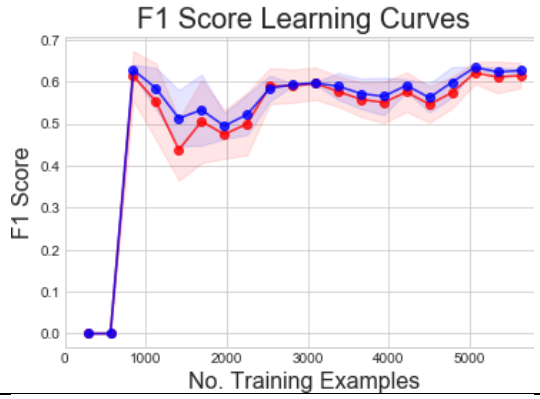
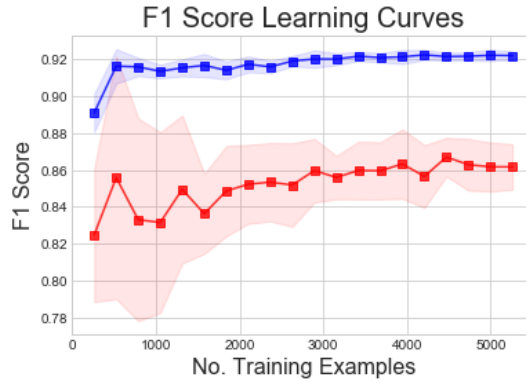
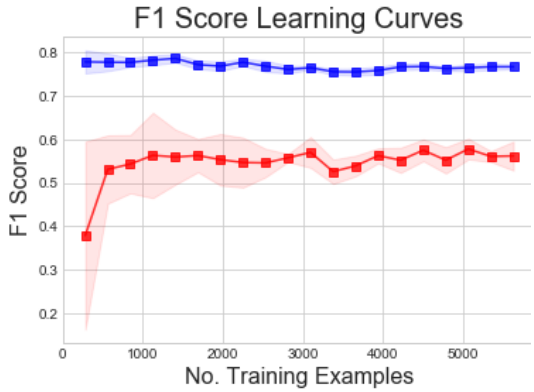
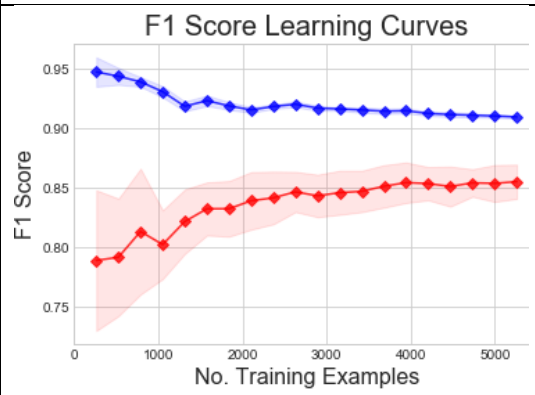
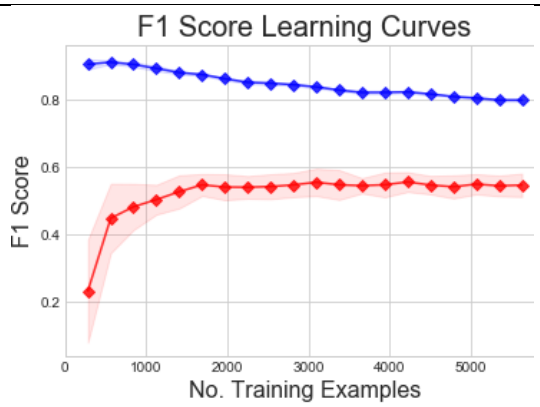
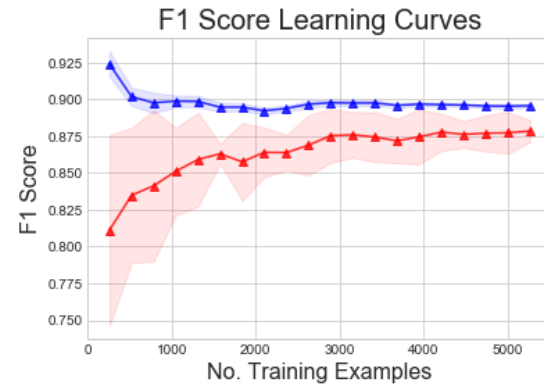
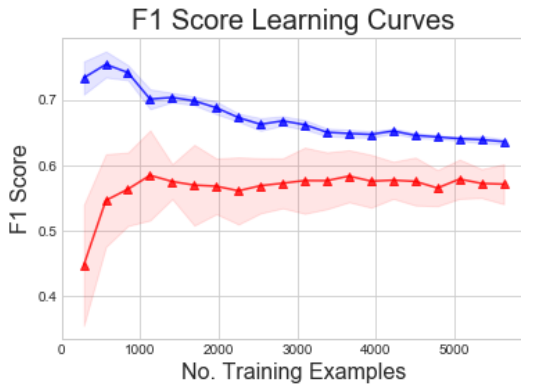
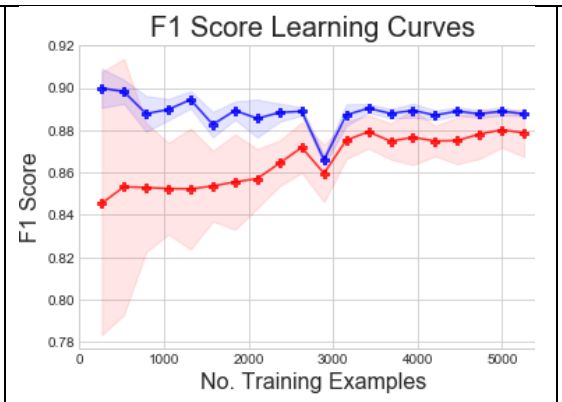
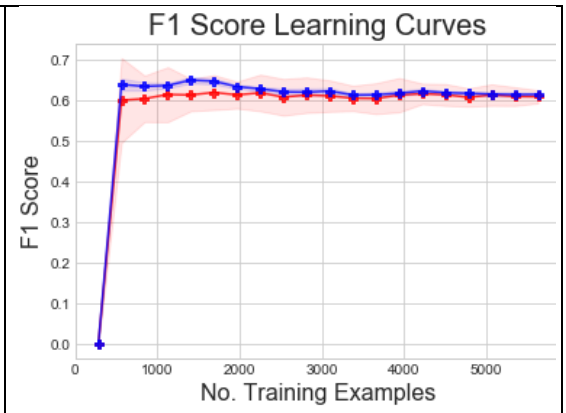
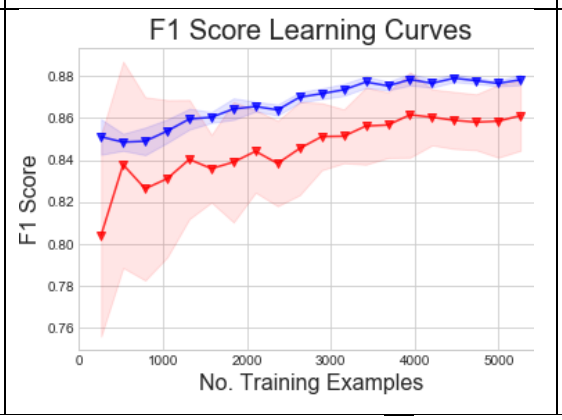
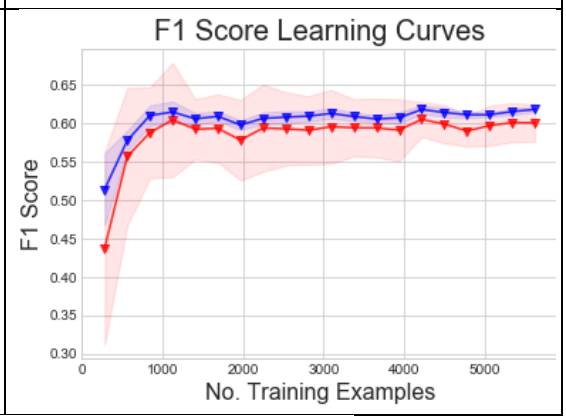


Table 1: Model complexity curves

Learning curves: CV scores

Algorithm	Dataset#1: Wind speed	Dataset#2: Customer retention
Decision Tree		
Boosting		
SVM (Kernel: poly)		
SVM (Kernel: rbf)		

Neural Network		
K Nearest Neighbors		

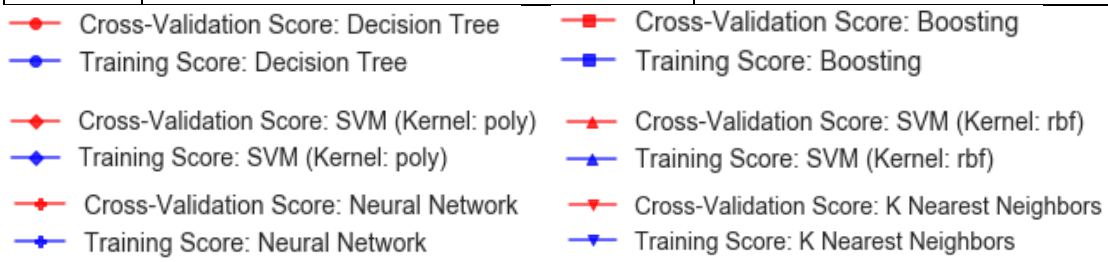
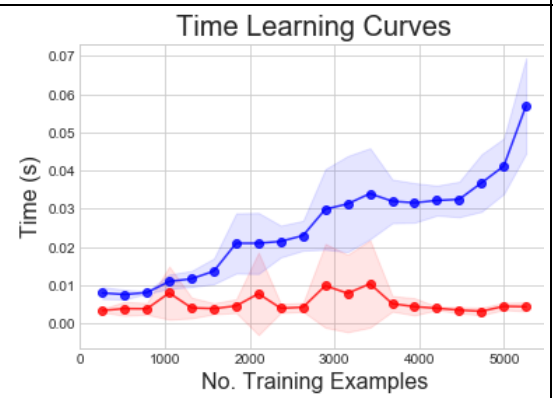
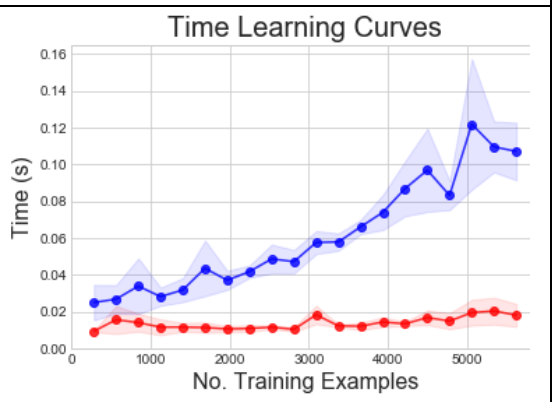
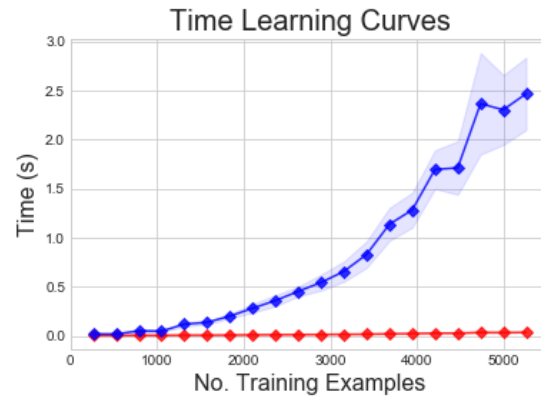
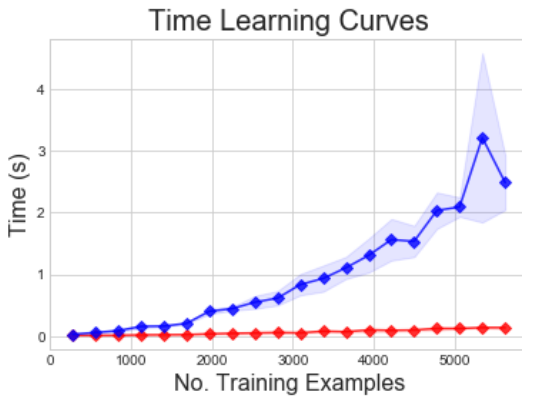
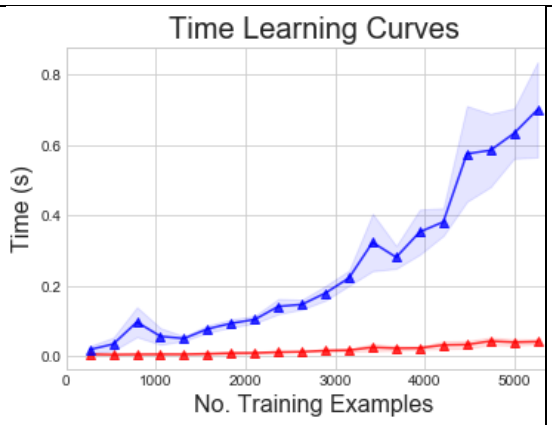
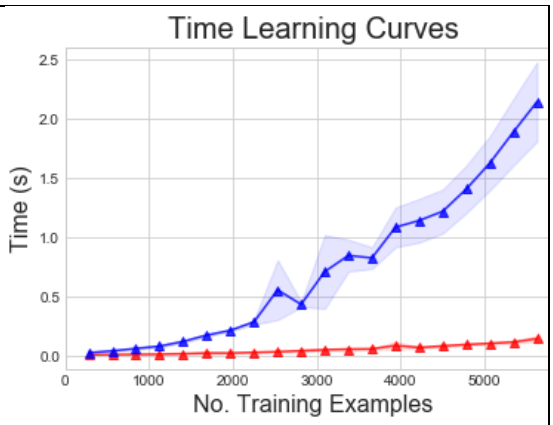
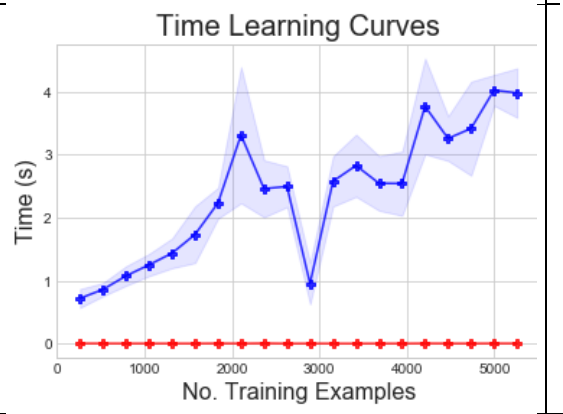
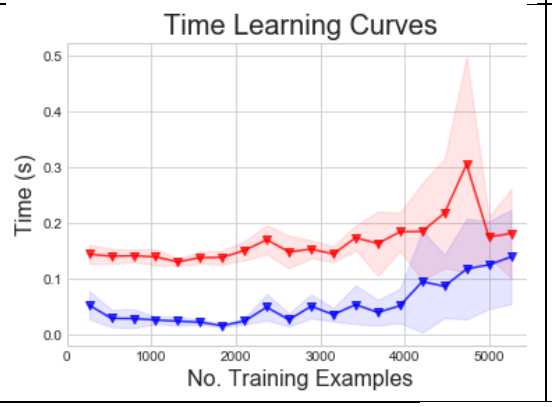
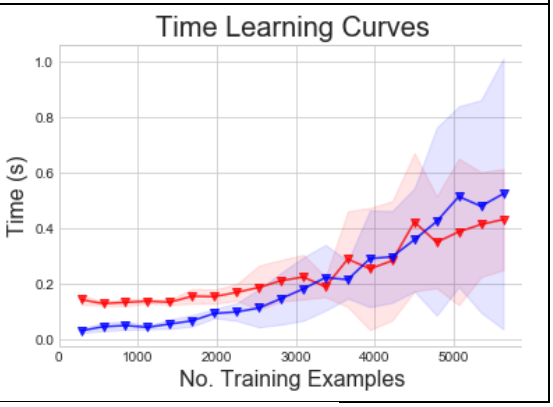


Table 2: Learning curves: CV scores

Learning curves: Training times

Algorithm	Dataset#1: Wind speed	Dataset#2: Customer retention
Decision Tree		
Boosting		
SVM (Kernel: poly)		

SVM (Kernel: rbf)		
Neural Network		
K Nearest Neighbors		

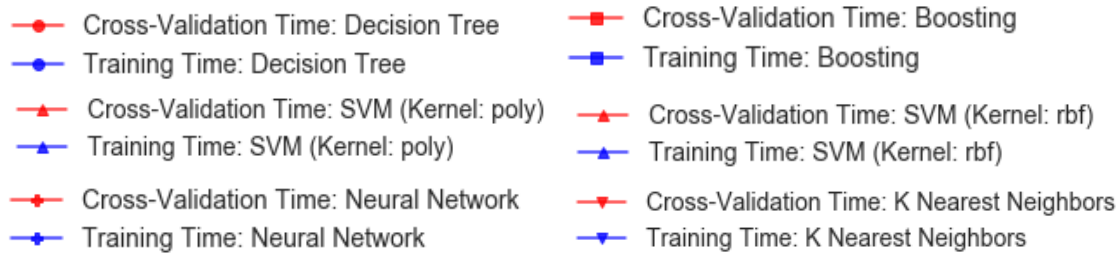


Table 3: Learning curves: Training times

Model performance on out-of-sample test data

Algorithm	AUC	F1	Accuracy	Precision	Recall
Decision Tree	0.833357	0.839089	0.83346	0.842183	0.836018
Boosting	0.845979	0.85277	0.846388	0.849057	0.856515
SVM (Kernel: poly)	0.833568	0.852437	0.836502	0.802326	0.909224
SVM (Kernel: rbf)	0.86127	0.867153	0.861597	0.864629	0.869693
Neural Network	0.859511	0.866473	0.860076	0.858993	0.874085
K Nearest Neighbors	0.852886	0.862464	0.853992	0.84432	0.881406

Table 4: Model performance on dataset#1: Wind speed

Algorithm	AUC	F1	Accuracy	Precision	Recall
Decision Tree	0.712017	0.578512	0.782516	0.596591	0.561497
Boosting	0.693461	0.552352	0.790334	0.638596	0.486631
SVM (Kernel: poly)	0.67705	0.525588	0.756219	0.544413	0.508021
SVM (Kernel: rbf)	0.68922	0.545181	0.785359	0.624138	0.483957
Neural Network	0.711141	0.578279	0.787491	0.61194	0.548128
K Nearest Neighbors	0.712409	0.577957	0.77683	0.581081	0.574866

Table 5: Model performance on dataset#2: Customer retention

Model comparison curves

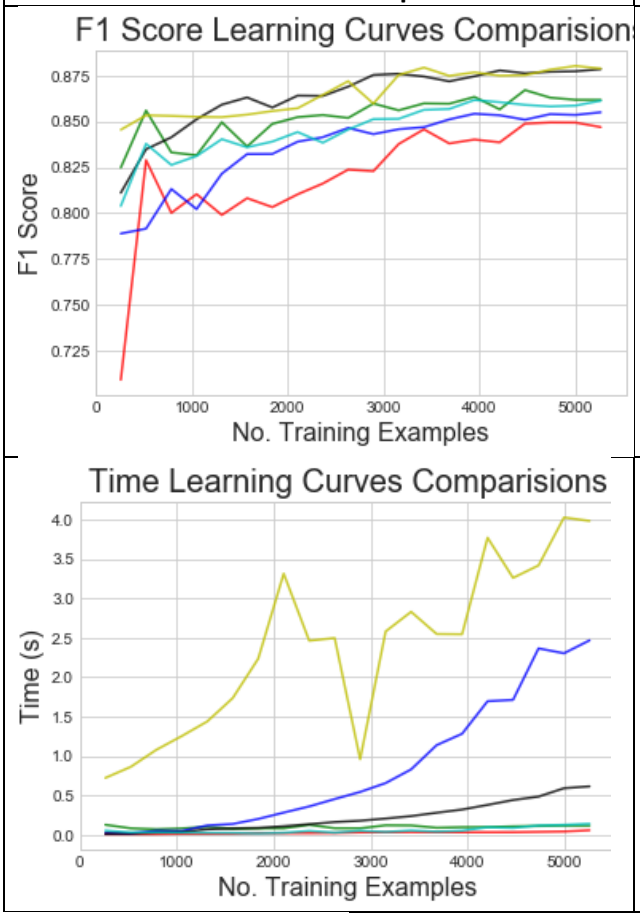
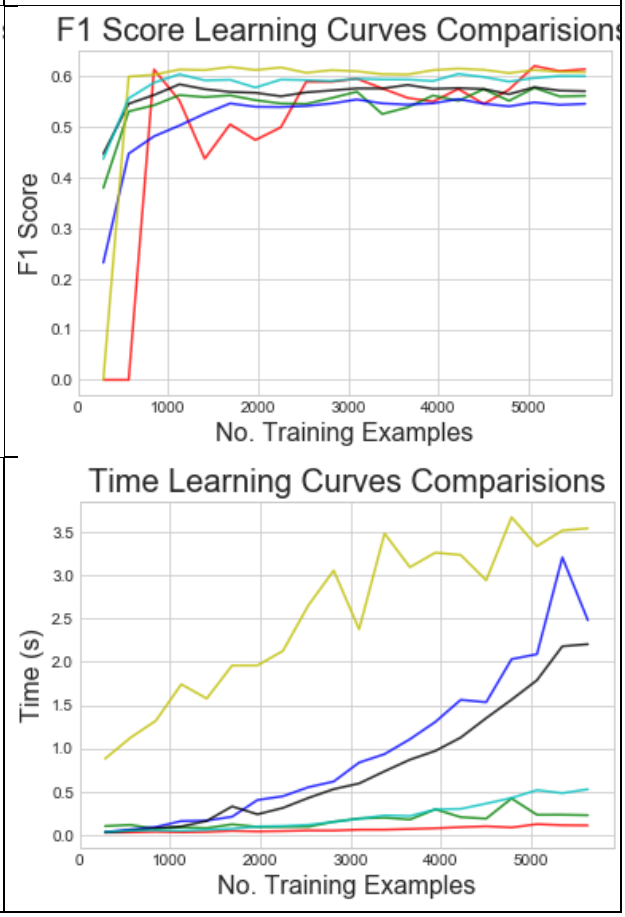
Dataset#1: Wind speed	Dataset#2: Customer retention
	

Table 6: Model comparison curves

Discussion

Following conclusions can be drawn after analyzing 6 different model on 2 data sets mentioned above:

- As model complexity increases, both training scores and CV scores improves. But later, as model becomes too complex and CV score decreases while training score keeps on improving. This is point where the model starts overfitting (Table. 1). This is true for all algorithms across both the datasets.
- Initially, CV score learning curves sharply rise indicating improvements resulted from new training instances. After a few 1000s of training instances a point is reached after which negligible improvements in CV scores is observed (Table. 2). This is true for all algorithms across both datasets.
- Training score learning curves indicates that model performance on instances used for training is consistently better than CV instances. This is due to fact model has knowledge training data and so it is expected perform better on it as compared to CV data (Table. 2). This is true for all algorithms across both datasets.
- The difference between CV score and training score learning curves is small and decreases as a greater number of training instances are added. If this difference is large or doesn't decrease with number of training instances, it would indicate overfitting (Table. 2). This is true for all algorithms across both datasets.
- Training time learning curves rise with number of training instances (Table 3). But the rate of increase is different for different algorithms. SVM (both poly and rbf kernels) and Nerural Network seems to be polynomial whereas other look like linear in number of training instances. This comparison is more evident in Table 6.
- CV time learning curves typically (except KNN) remains flat indicating that prediction is mostly independent of number of training instances (Table 3). This could be attributed to the fact that most of these algorithms (except KNN) learn some sort of internal state (coefficients) and prediction involves applying this state on the input making it independent of training size.
- CV time learning curve seems to rise proportionally to number of training instances. Additionally, CV time required for KNN is typically more than training time (Table 3). This could be because KNN works differently; it memorizes the training examples during training and during prediction it goes over them to find K-nearest neighbors. And so, more the number of training examples prediction more expensive.
- As a greater number of training instances are added, the algorithm used becomes irrelevant. This is evident from the fact that the CV scores and model performance on unseen data is almost identical for all the algorithms across both the datasets (Table 4, Table 5 and Table 6).
- Hyperparameters tuned by 10-fold CV lead to good generalization over unseen data, as model performance over unseen data for each model is almost same as CV score for that model (Table 2, Table 4 and Table 5).

To further improve model performance following options can be explored:

- Feature selection and dimensionality reduction can help in improving model performance.
- Better hyperparameters can be found by doing more exhaustive grid search over multiple hyperparameters. But doing so would be time consuming.