Analysis_Final2

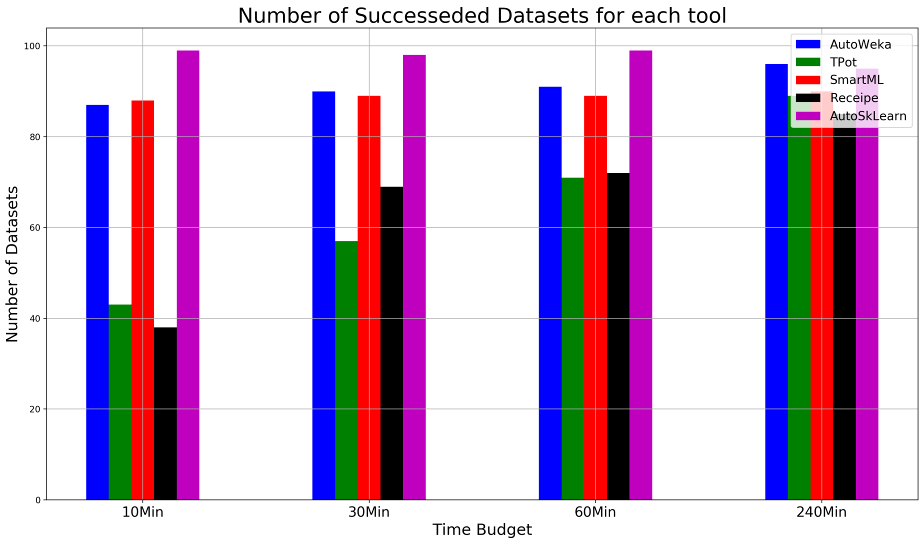
June 17, 2019

```
In [2]: import math
        from random import *
        import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
        import seaborn as sns
        from IPython.display import Image
        from IPython.core.display import HTML
```

0.1 1- Number of datasets succeeded by each tool in each time budget

```
In [3]: Image("Success.png")
```

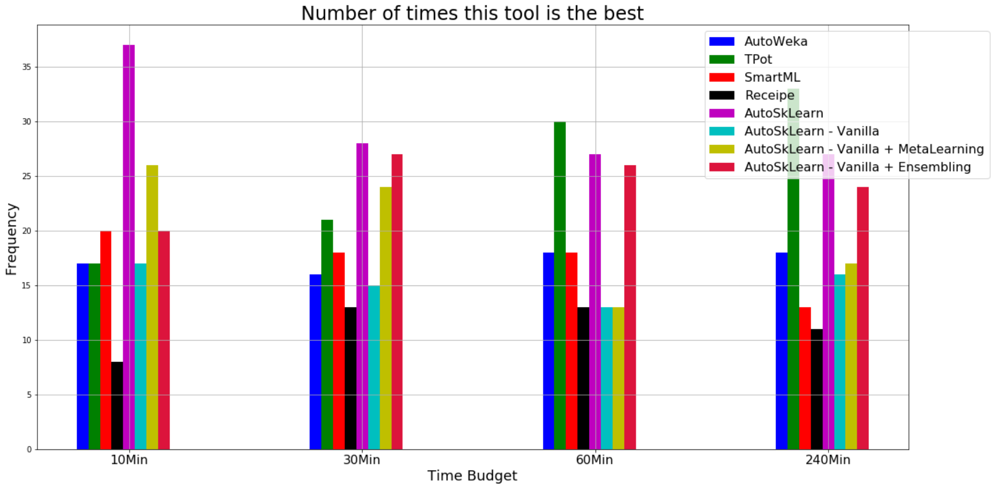
```
Out[3]:
```



0.2 2- How many times each tool has been the best performance?

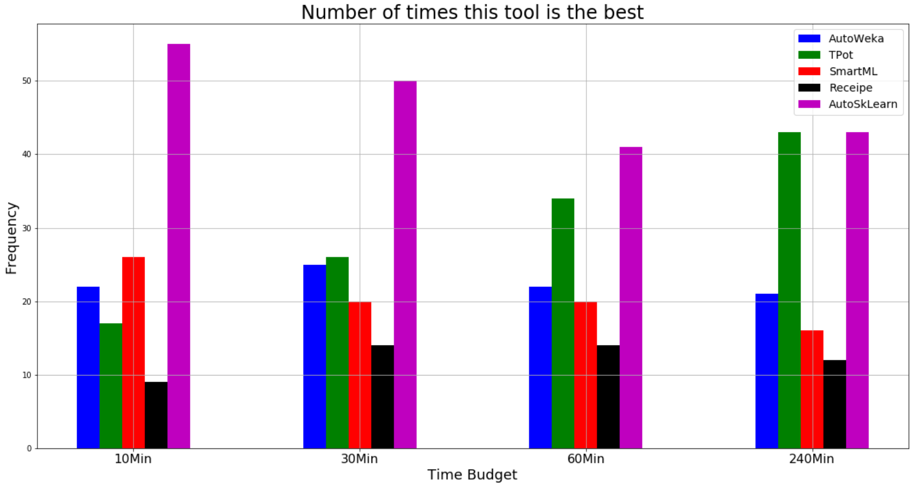
```
In [4]: Image('Best_All.png')
```

```
Out[4]:
```



```
In [5]: Image('Best_Main.png')
```

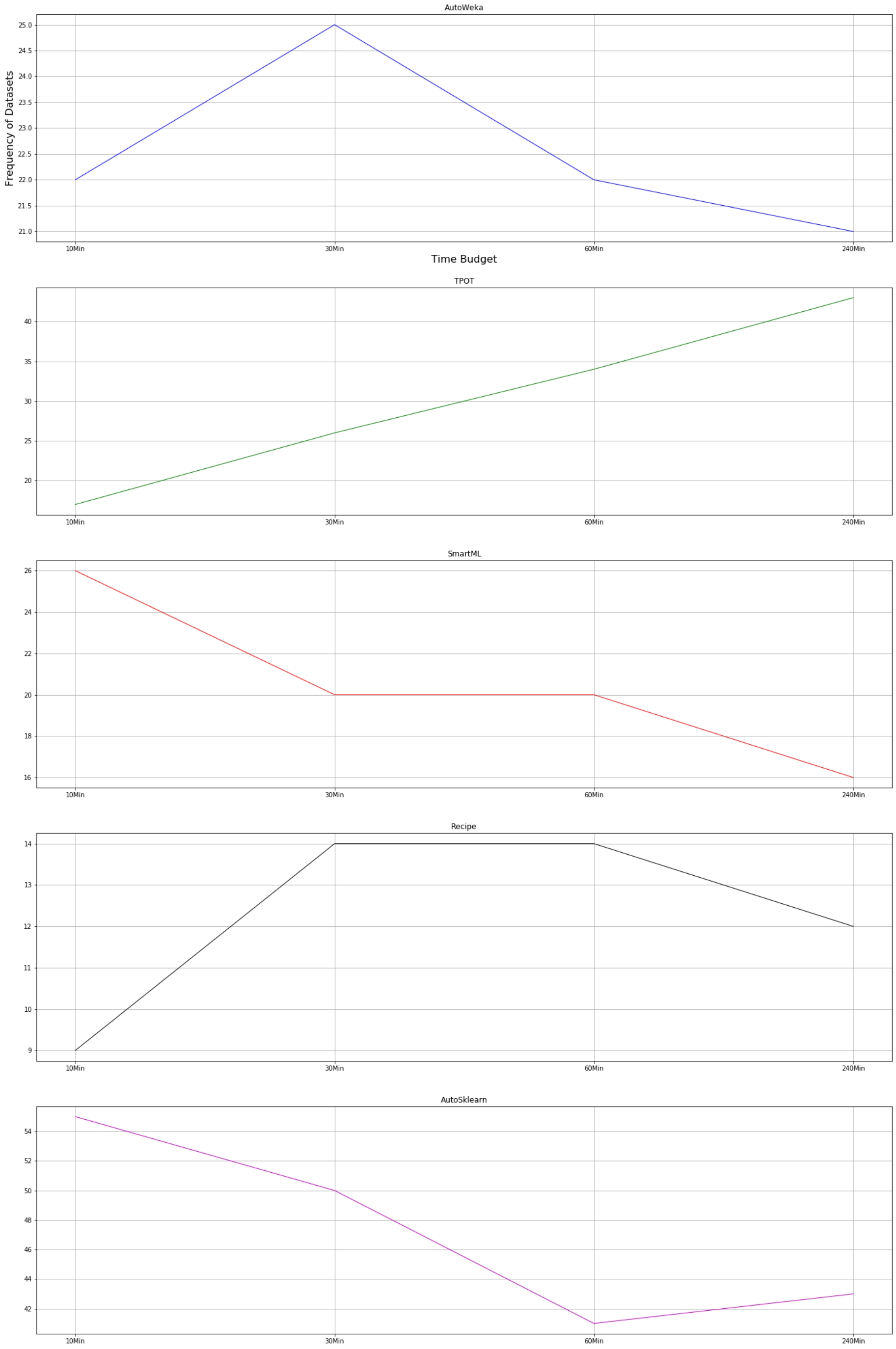
```
Out[5]:
```



```
In [28]: Image('Diff_MetaLearning2.png')
```

```
Out[28]:
```

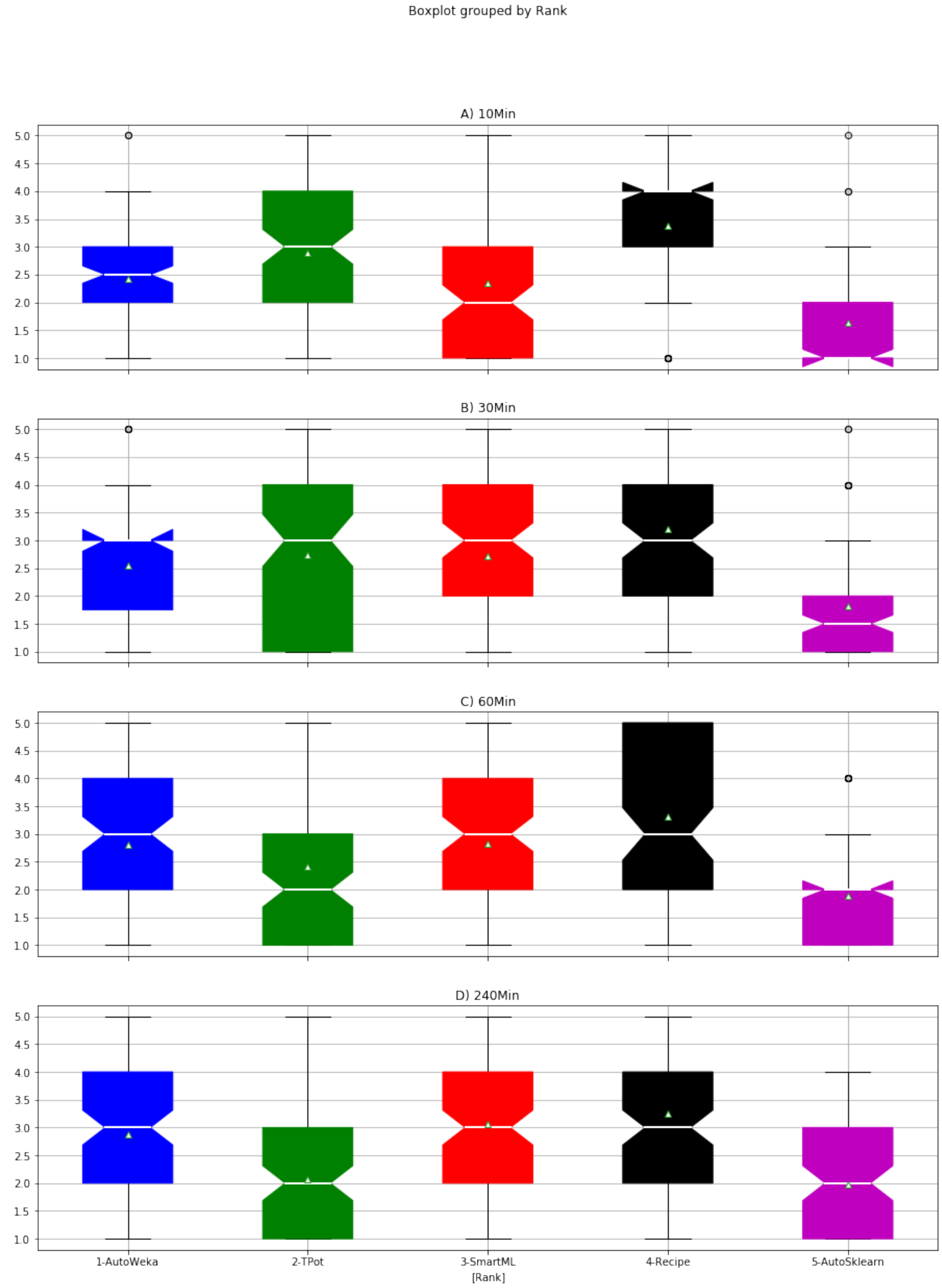
Numbre of time best performance fro each tool separately



0.3 3- What is the average ranking for each tool?

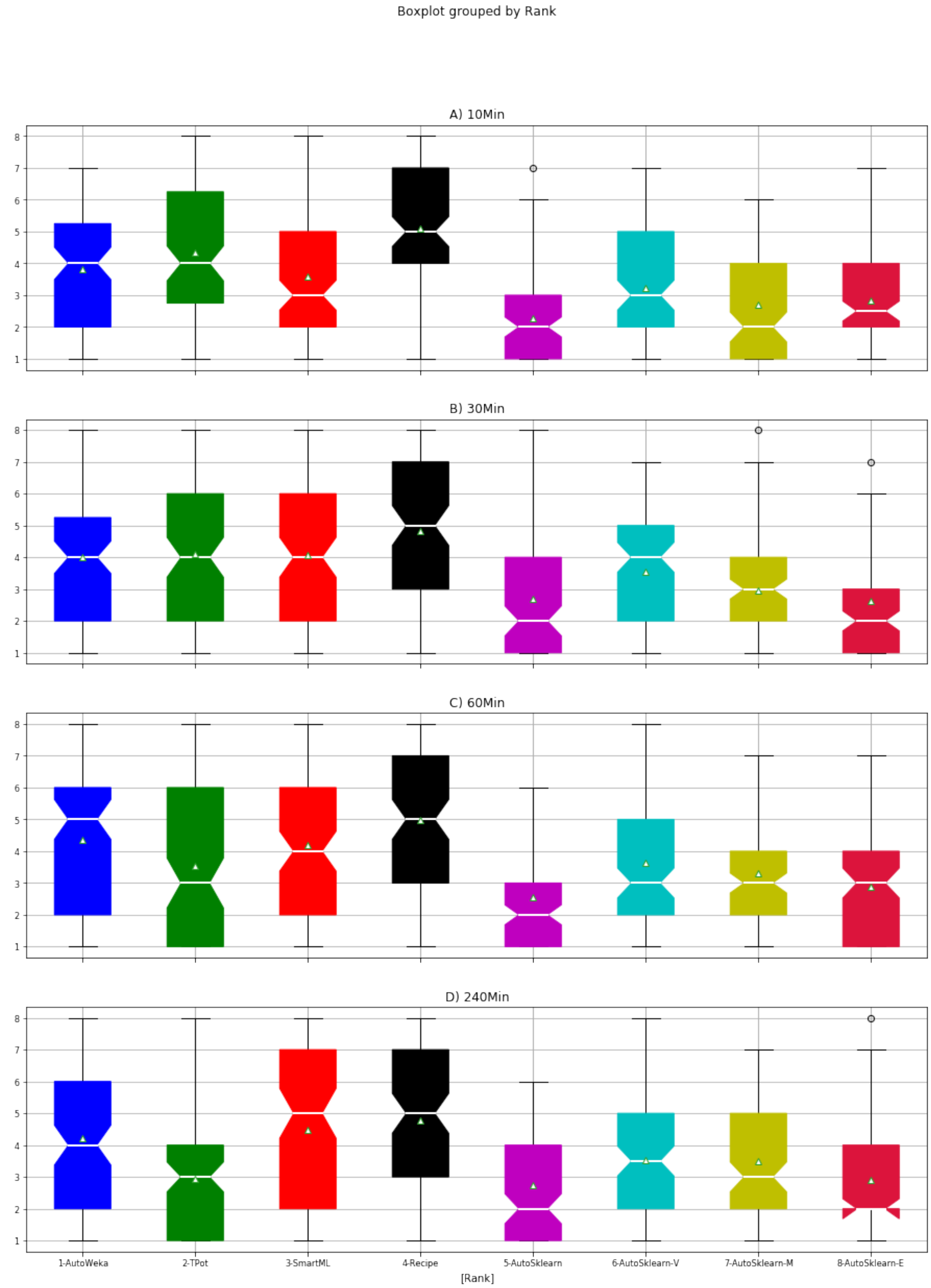
```
In [11]: Image('AvgRank_Main.png')
```

```
Out[11]:
```



```
In [10]: Image('AvgRank_All.png')
```

Out[10]:

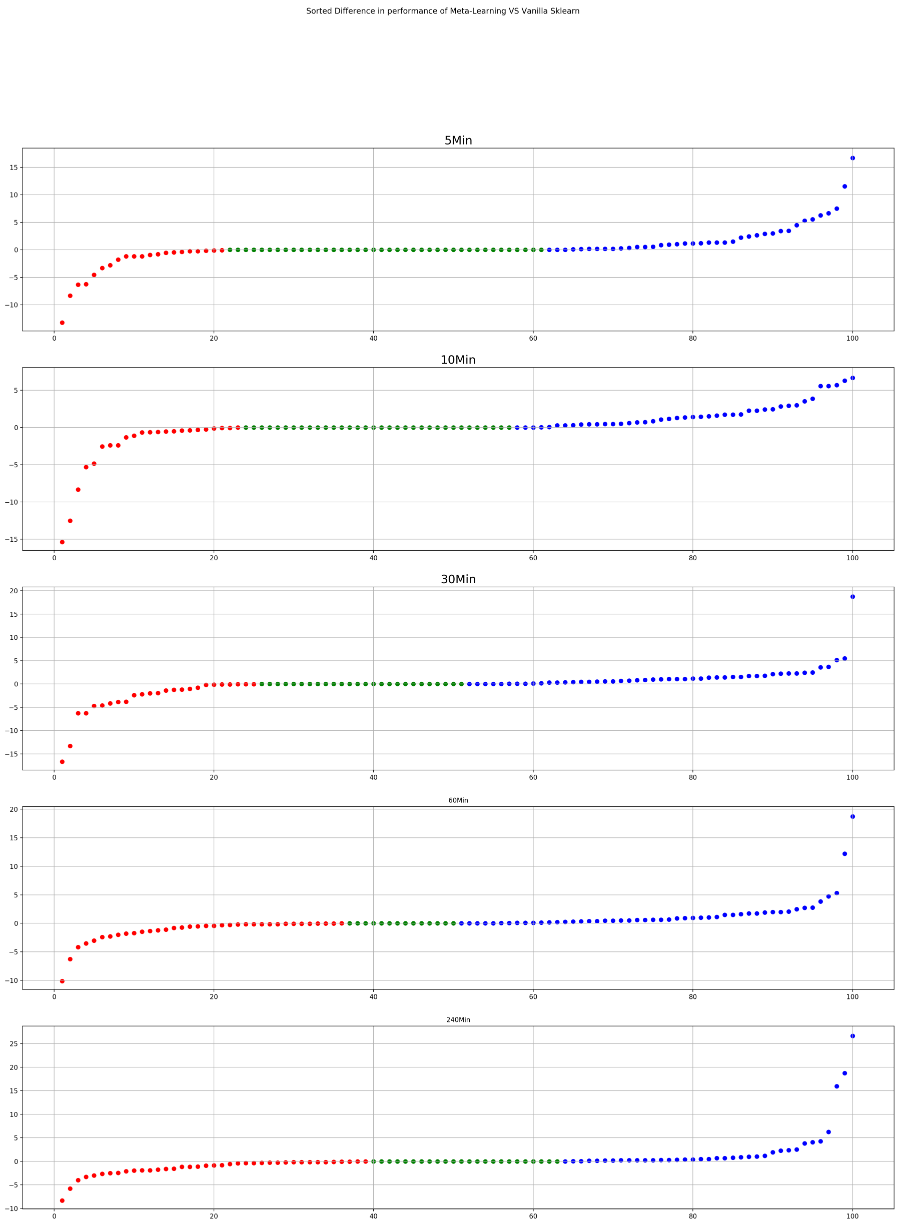

Boxplot grouped by Rank
A) 10Min
B) 30Min
C) 60Min
D) 240Min
1
2
3
4
5
6
7
8
1-AutoWeka
2-TPot
3-SmartML
4-Recipe
5-AutoSklearn
6-AutoSklearn-V
7-AutoSklearn-M
8-AutoSklearn-E
[Rank]

0.4 4- How many times Meta-learning improved the Performance and what is the average improvement, in scikit learn?

0.5 5- How many times Meta-learning performed lower in scikit learn?

```
In [12]: Image('Diff_MetaLearning.png')
```

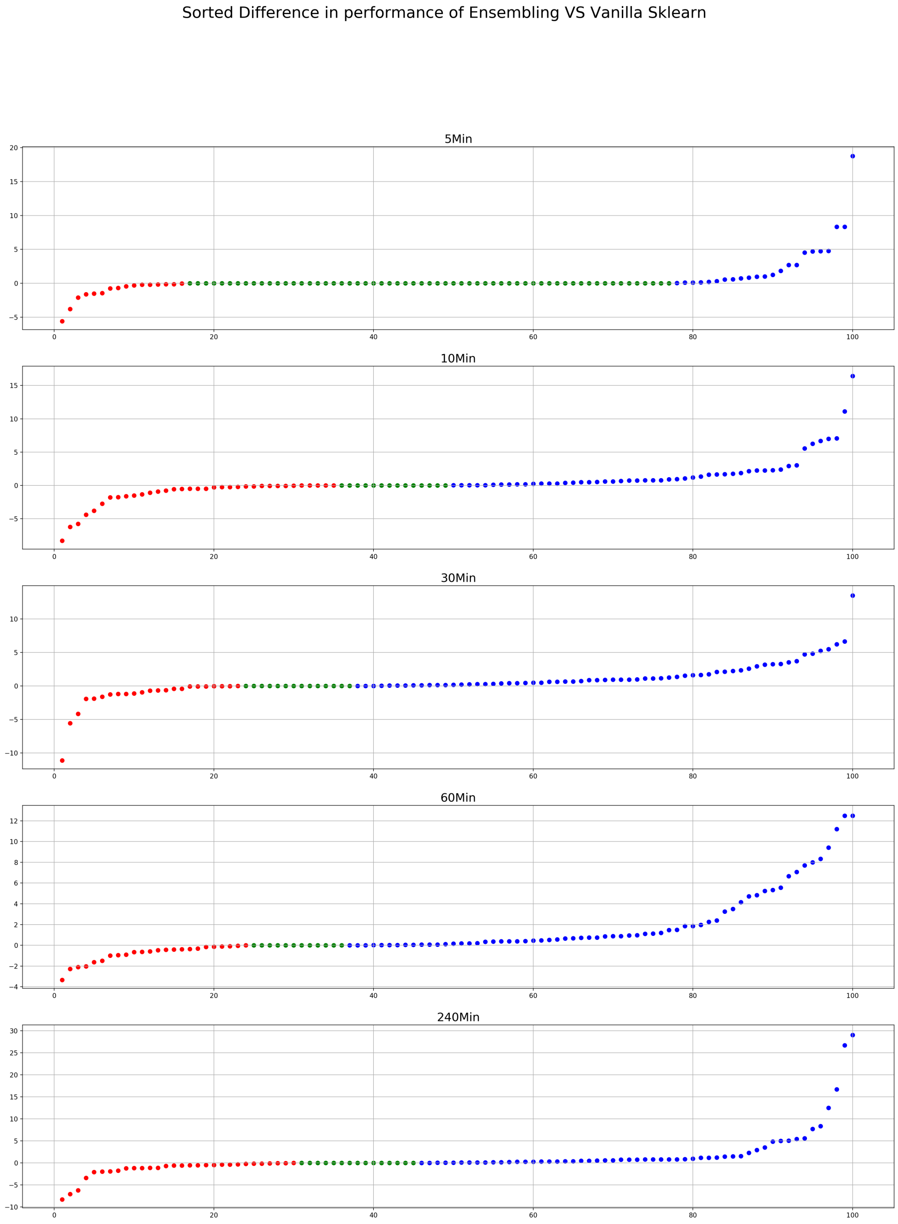
```
Out[12]:
```



0.6 6- Same for ensembling

```
In [13]: Image('Diff_Ensembling.png')
```

```
Out[13]:
```



0.7 7- Comparison between TPOT and recipe ONLY (Performance and Search Space)ű

0.7.1 TPOT Search Space:

Classifiers (11)

Preprocessors & Extractors (14)

Feature Selectors (5)

0.7.2 Recipe Search Space:

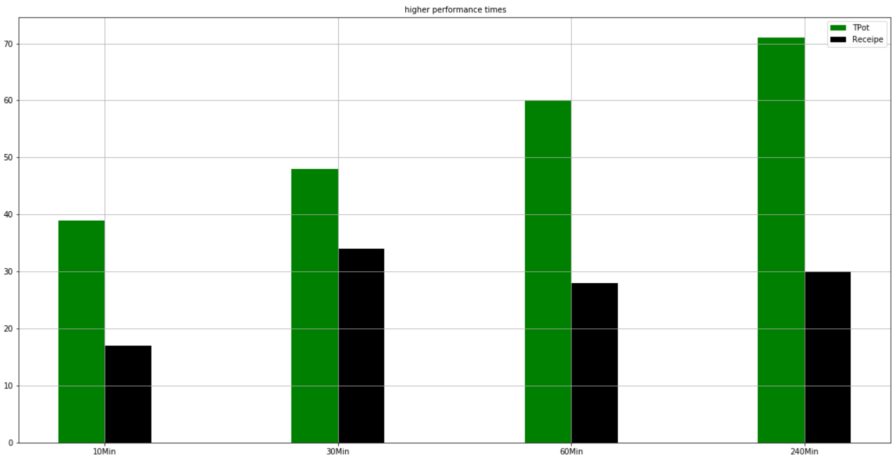
Classifiers (20)

Preprocessors & Extractors (15)

Feature Selection (8)

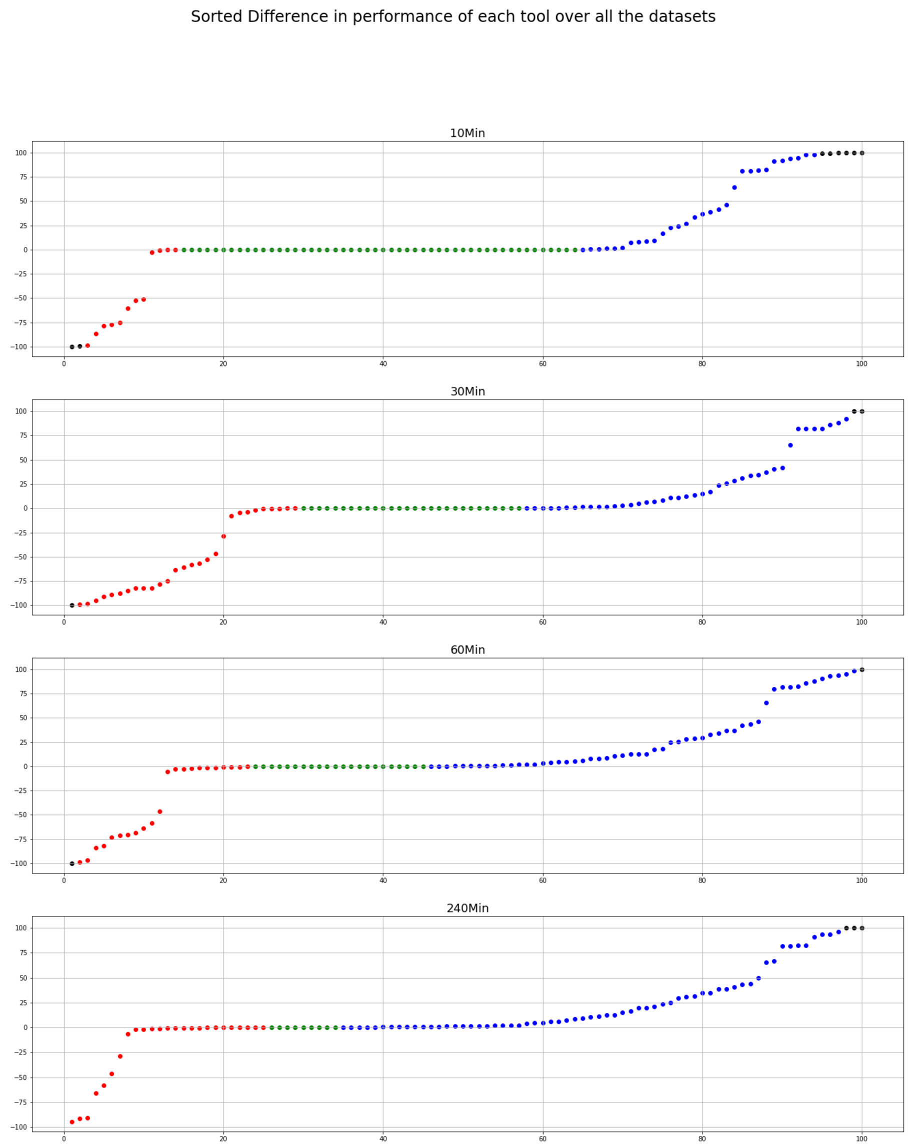
```
In [14]: Image('Best_Genetic.png')
```

```
Out[14]:
```



```
In [15]: Image('Diff_Genetic.png')
```

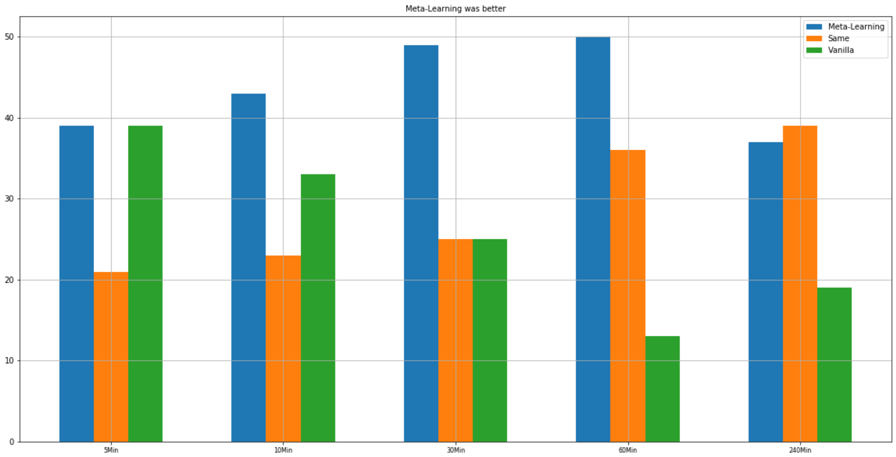
```
Out[15]:
```



0.8 8- Does the impact of Meta-learning increase or decrease by more time?

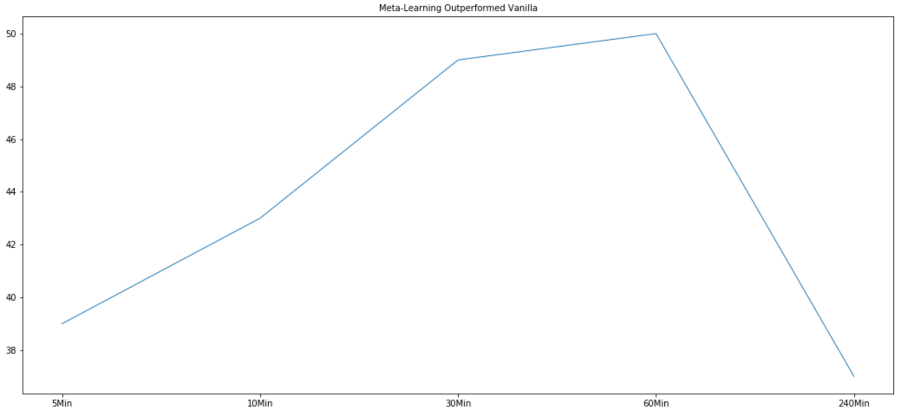
```
In [16]: Image('Meta_Best_Worse.png')
```

```
Out[16]:
```



```
In [17]: Image('MetaImpact.png')
```

```
Out[17]:
```

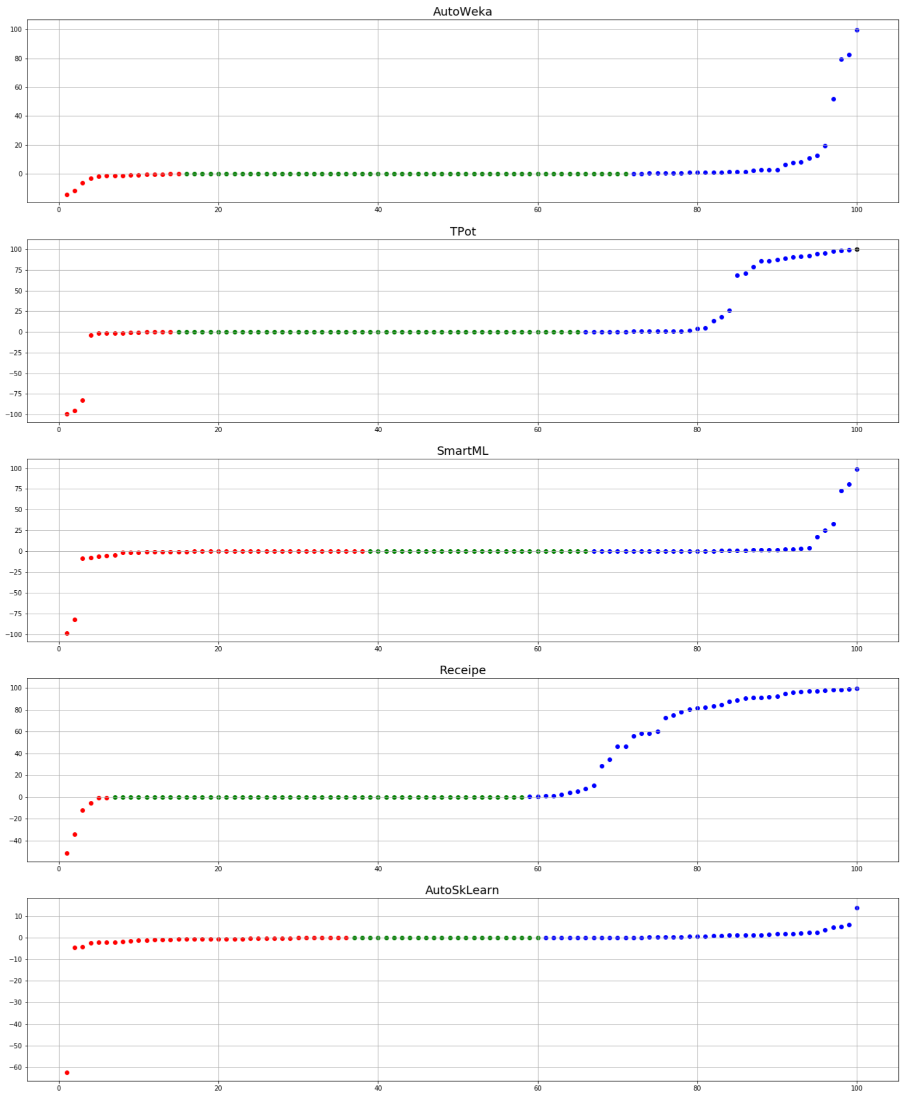


0.9 9. Performance Gain by increasing time budget

```
In [18]: Image('Diff_Main_10_30.png')
```

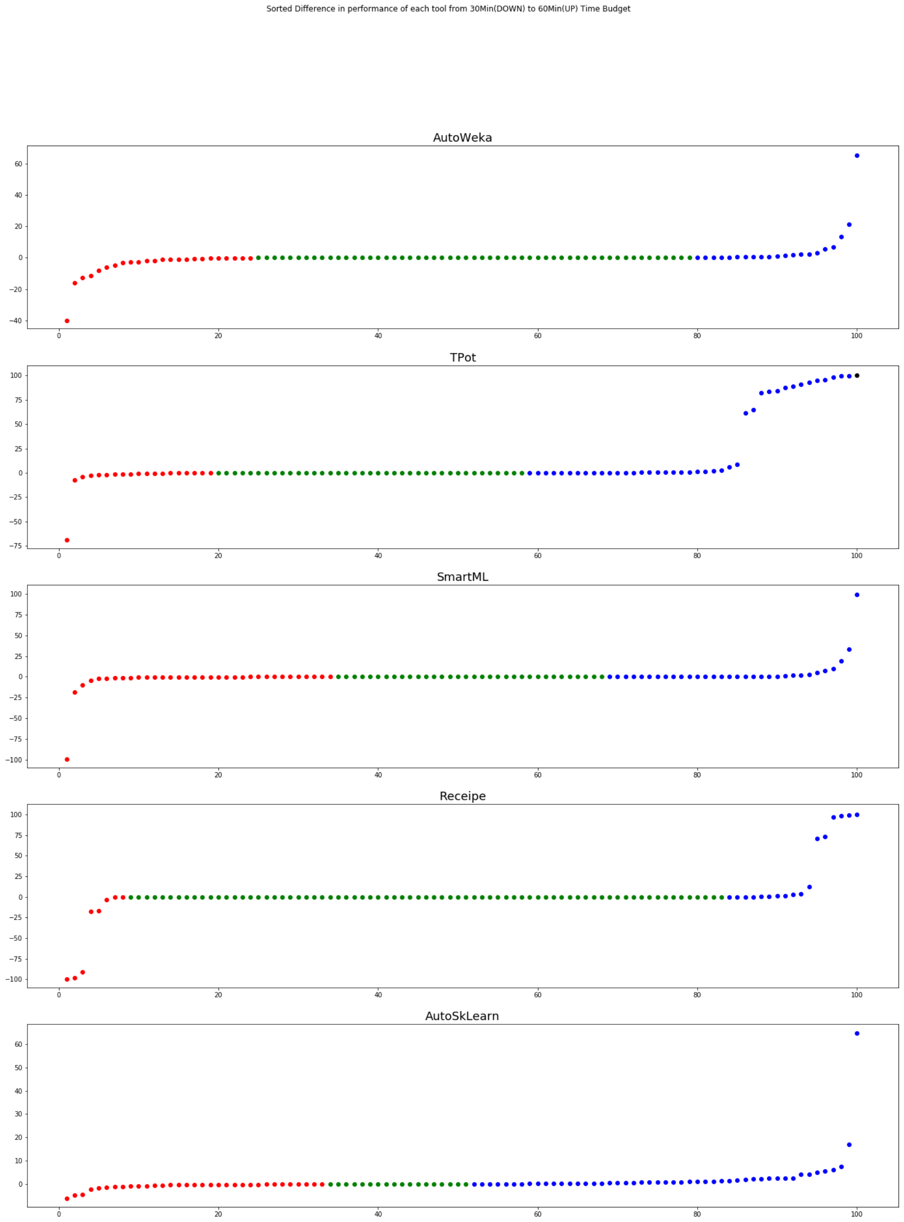
```
Out[18]:
```

Sorted Difference in performance of each tool from 10Min(DOWN) to 30Min(UP) Time Budget



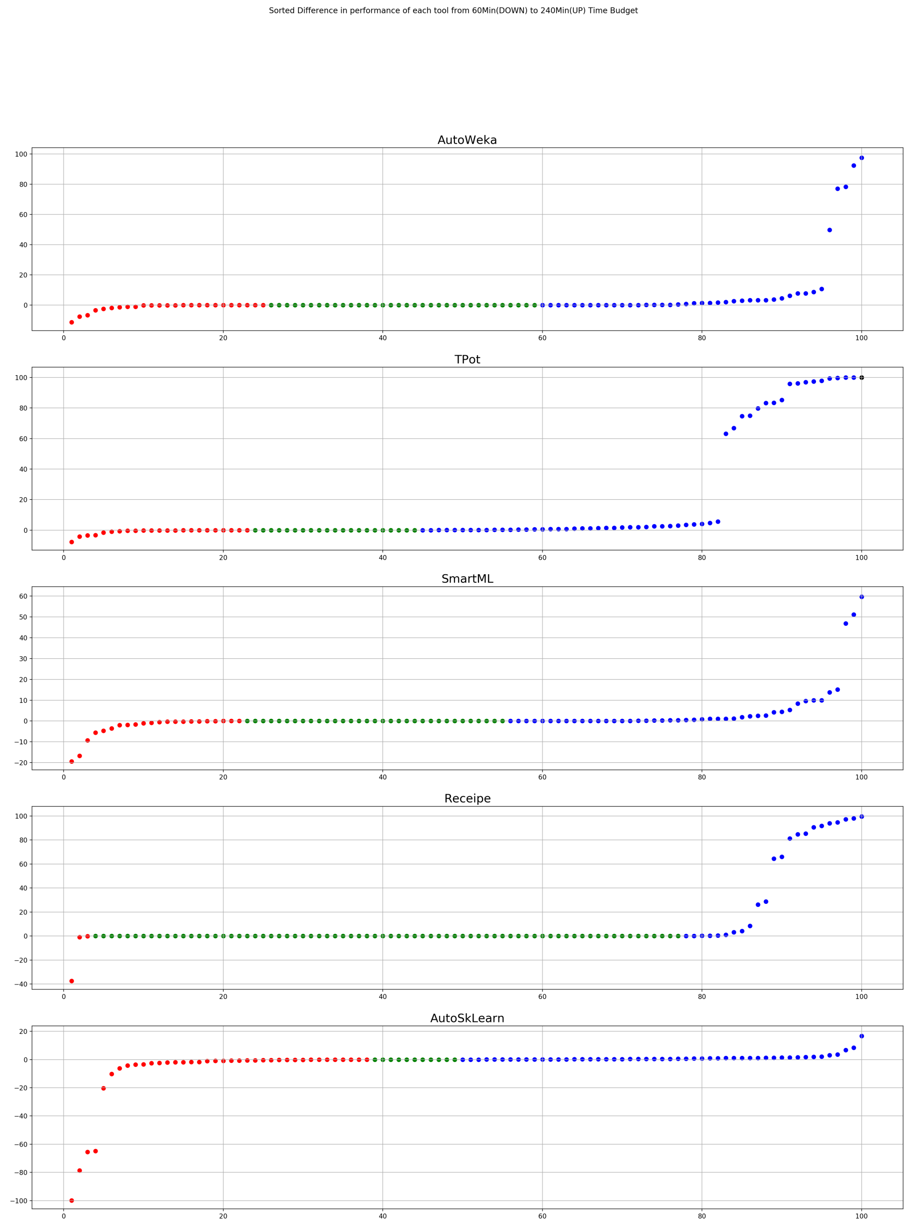
```
In [19]: Image('Diff_Main_30_60.png')
```

```
Out[19]:
```



```
In [20]: Image('Diff_Main_60_240.png')
```

```
Out[20]:
```



0.10 10. Output pipelines for datasets where TPOT outperform AutoSklearn

************ CHURN DATASET ************ TPOT: Pipeline(memory=None,steps=[('extratreesclassifier')])

AutoSklearn: {'balancing:strategy': 'none', 'categorical_encoding:**choice**': 'one_hot_encoding', 'classifier:**choice**': 'random_forest', 'imputation:strategy': 'mean', ‘preprocessor:**choice**’: ‘no_preprocessing’, ‘rescaling:**choice**’: ‘standardize’, 'categorical_encoding:one_hot_encoding}

************ Adult DATASET ************ TPOT: Pipeline(memory=None, steps=[(‘gradientboostingclassifier’)])

AutoSklearn: {‘balancing:strategy’: ‘none’, ‘categorical_encoding:**choice**’: ‘one_hot_encoding’, ‘classifier:**choice**’: ‘random_forest’, ‘imputation:strategy’: ‘median’, ‘preprocessor:**choice**’: ‘feature_agglomeration’, ‘rescaling:**choice**’: ‘robust_scaler’, 'categorical_encoding:one_hot_encoding}

************ Yeast DATASET ************ TPOT: Pipeline(memory=None, steps=[(‘randomforestclassifier’)])

AutoSklearn: {‘balancing:strategy’: ‘none’, ‘categorical_encoding:**choice**’: ‘one_hot_encoding’, ‘classifier:**choice**’: ‘extra_trees’, ‘imputation:strategy’: ‘median’, ‘preprocessor:**choice**’: ‘feature_agglomeration’, ‘rescaling:**choice**’: ‘normalize’, 'categorical_encoding:one_hot_encoding }

************ Eucalputus DATASET ************ TPOT: Pipeline(memory=None, steps=[(‘polynomialfeatures’, ‘gradientboostingclassifier’)])

AutoSklearn: ({‘balancing:strategy’: ‘none’, ‘categorical_encoding:**choice**’: ‘no_encoding’, ‘classifier:**choice**’: ‘extra_trees’, ‘imputation:strategy’: ‘most_frequent’, ‘preprocessor:**choice**’: ‘polynomial’, ‘rescaling:**choice**’: ‘normalize’

************ Ecoli DATASET ************ TPOT: Pipeline(memory=None, steps=[(‘featureunion’, transformer_list=[(‘rbfsampler’, ‘functiontransformer’)])

AutoSklearn: {‘balancing:strategy’: ‘weighting’, ‘categorical_encoding:**choice**’: ‘no_encoding’, ‘classifier:**choice**’: ‘libsvm_svc’, ‘imputation:strategy’: ‘median’, ‘preprocessor:**choice**’: ‘random_trees_embedding’, ‘rescaling:**choice**’: ‘standardize’}

0.10.1 AutoSklearn Search Space:

Classifiers (11):

1. Adaboost
2. BernoulliNB
3. GaussianNB
4. DecisionTreeClassifier
5. ExtraTreesClassifier
6. RandomForestClassifier
7. GradientBoostingClassifier
8. KNeighborsClassifier
9. svm.LinearSVC and non linear
10. LogisticRegression
11. XGBClassifier
12. LDA
13. Multinomial Naive-Bayes
14. SGD
15. QDA
16. Passive Aggressive

Preprocessors & Extractors (16):

1. Densifier

2. FastICA
3. cluster.FeatureAgglomeration
4. ExtraTrees
5. KernelPCA
6. Normalizer
7. Nystroem Sampler
8. PCA
9. PolynomialFeatures
10. Select Percentile
11. OneHotEncoder
12. Kitchen Sink
13. Imputation
14. Balancing
15. Variance Threshold
16. Rescaling

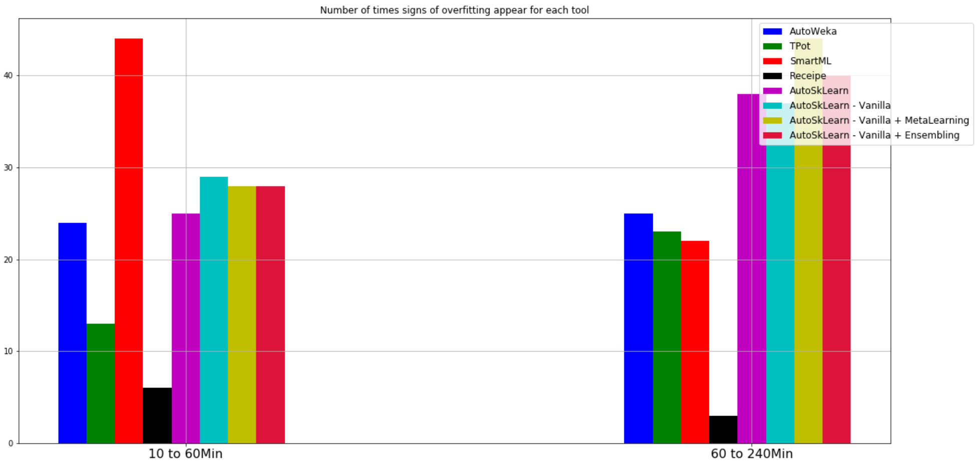
Feature Selectors (1):

1. truncated SVD

0.11 11- How models are over-fitted for each tool over time budget?

```
In [22]: Image('Overfit_All.png')
```

Out[22]:



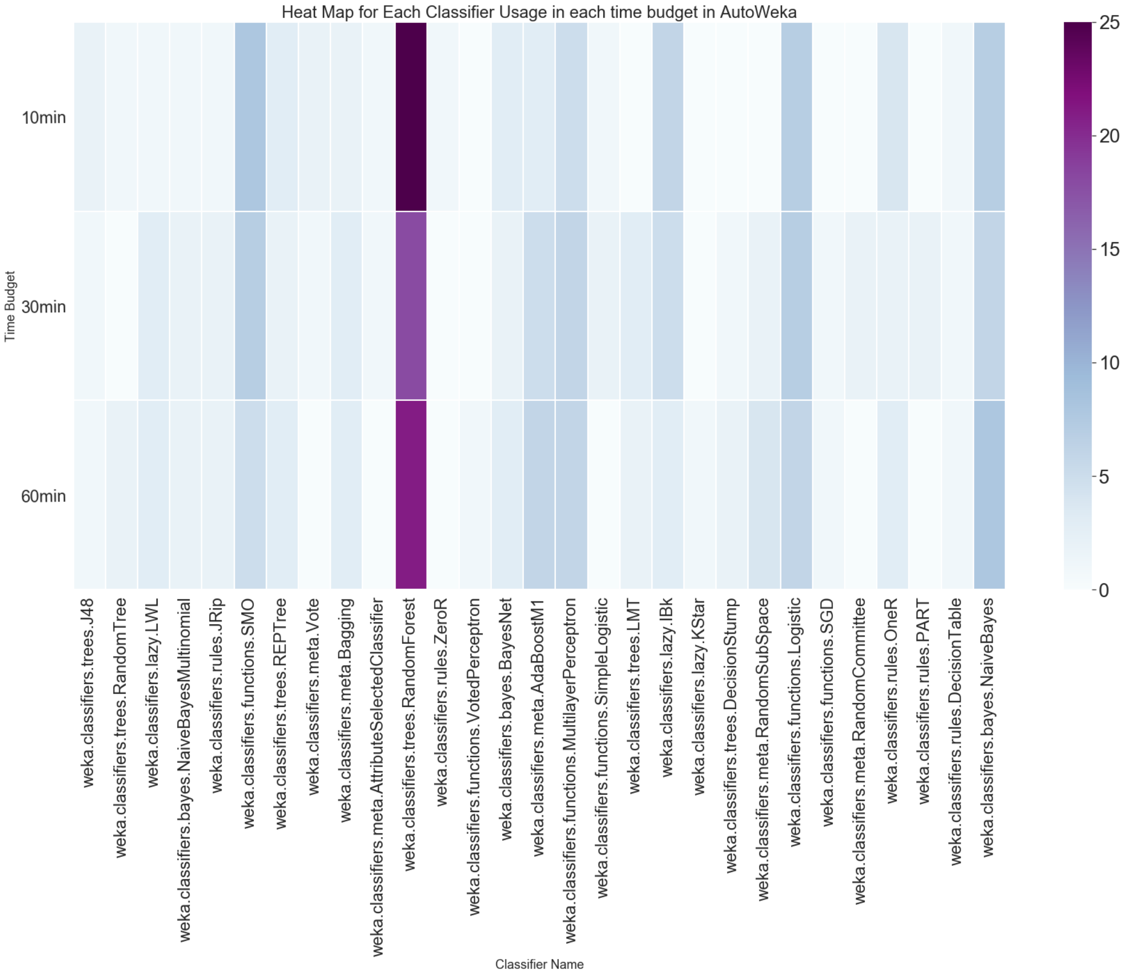
0.12 12- Try to correlate the above points with the datasets meta-features

Unfortunately, all correlations with meta-features are weak to moderate correlations according to spearman rank coefficient

0.13 13- Most frequent classifiers for each tool

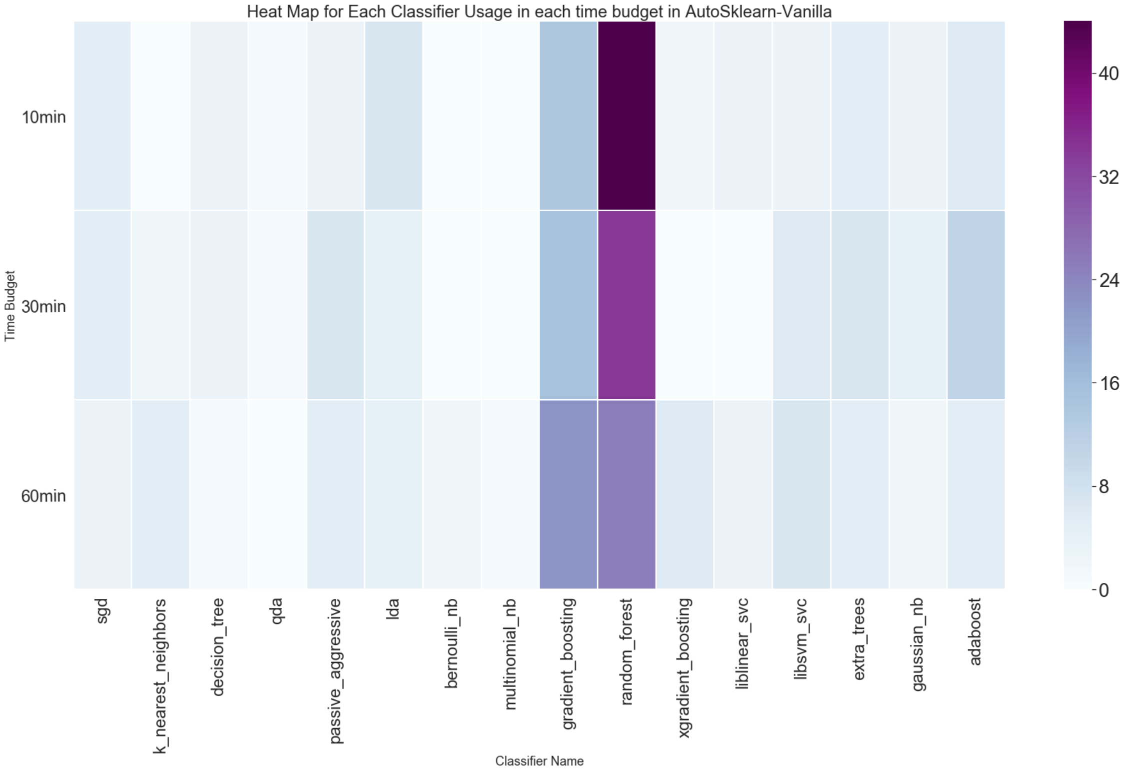
In [23]:
```
#### Most frequesnt classifiers chosen by Weka
Image('weka_most.png')
```

Out[23]:



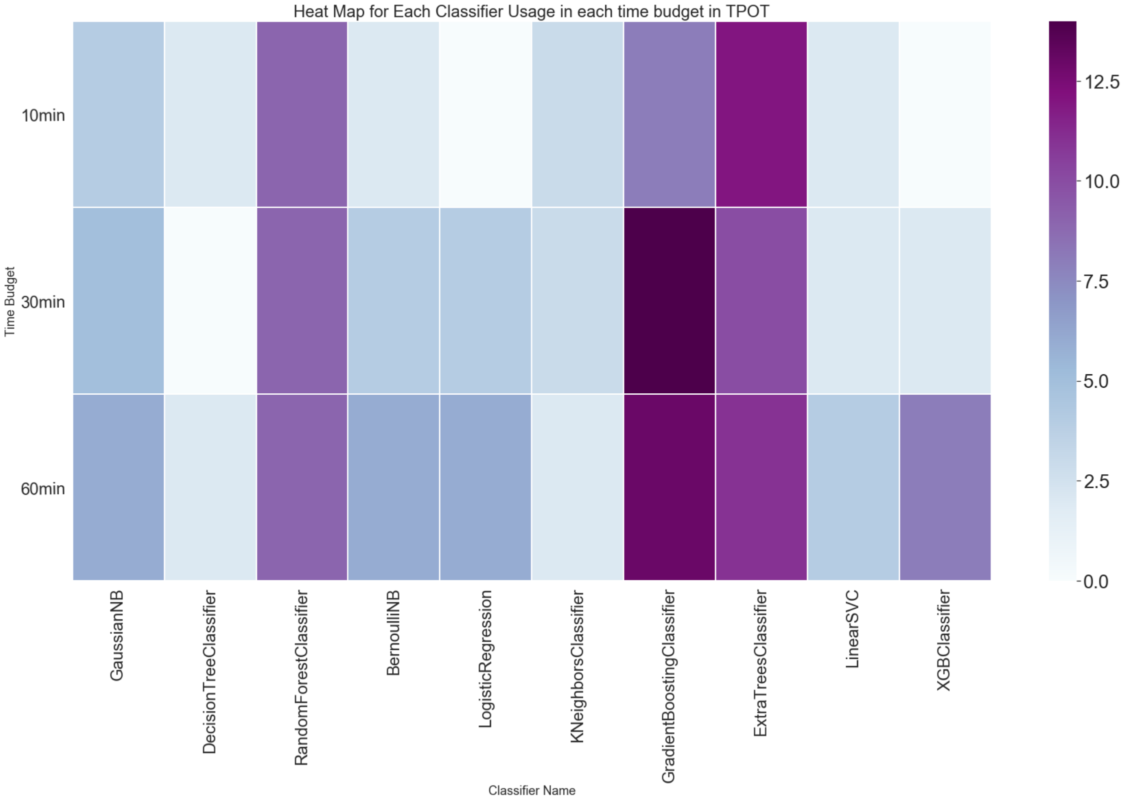
In [24]:
```
#### Most frequesnt classifiers chosen by AutoSklearn
Image('sklearn_most.png')
```

Out[24]:



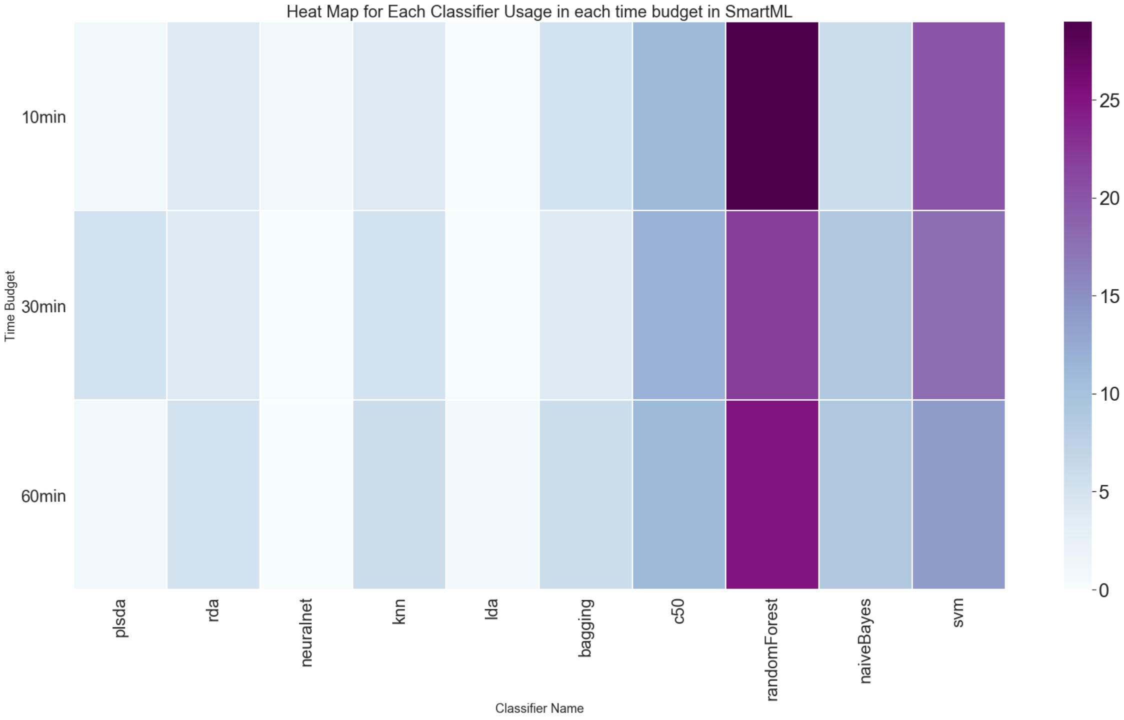
```
In [25]: #### Most frequesnt classifiers chosen by TPOT
         Image('tpot.png')

Out[25]:
```



```
In [26]: #### Most frequesnt classifiers chosen by SmartML
         Image('smartml.png')
```

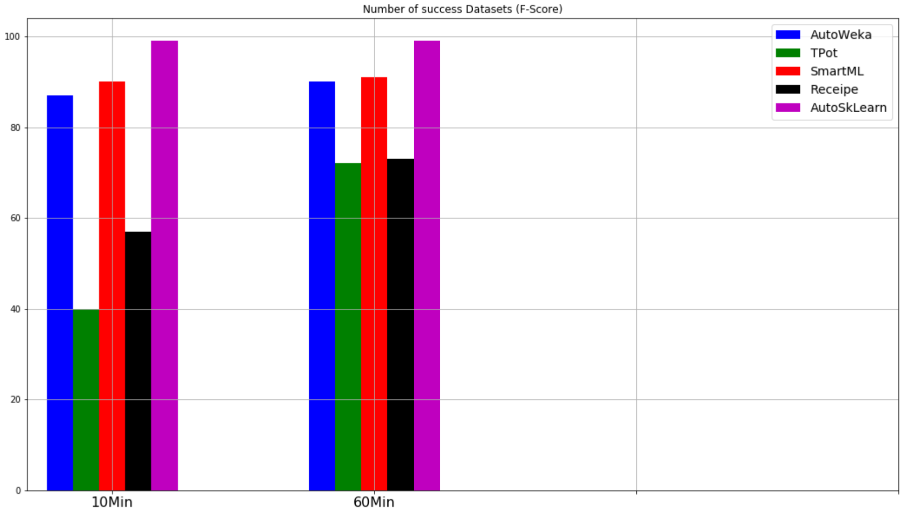
```
Out[26]:
```



0.14 14- FScore - Analysis

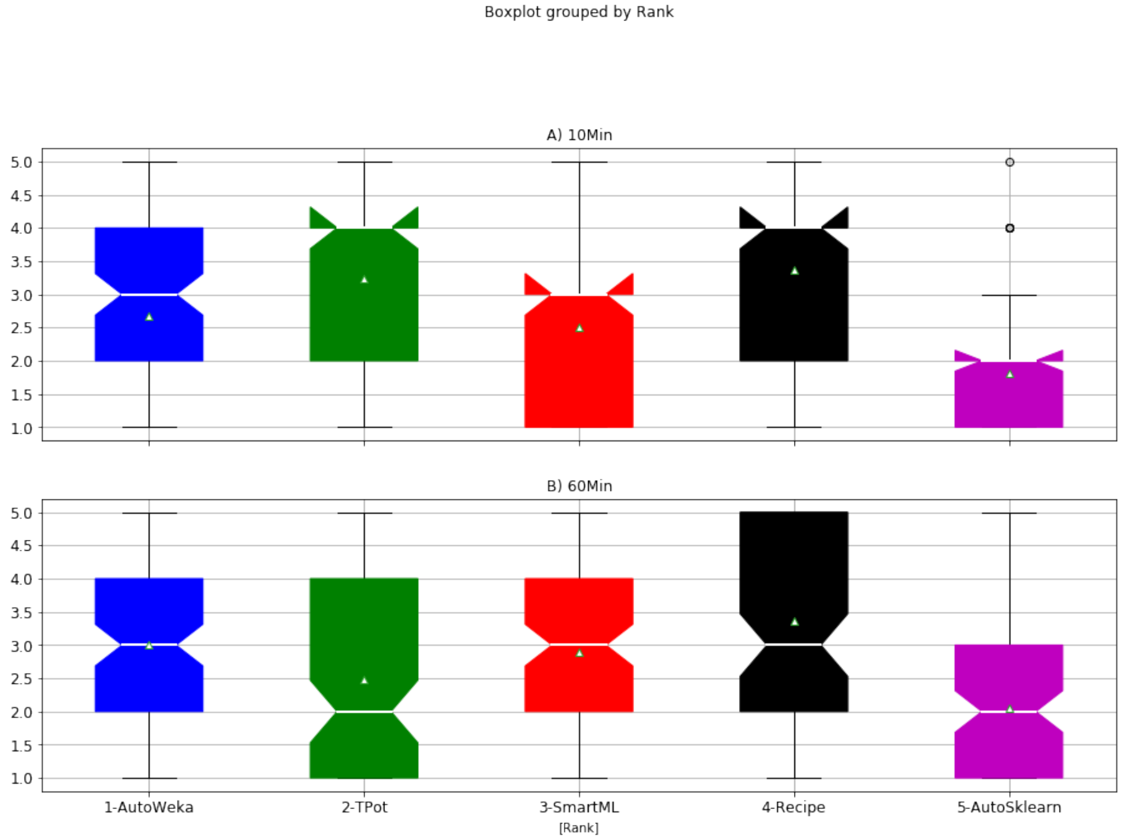
```
In [29]: # Number of Succeeded datasets for each tool
         Image('SuccessF.png')
```

```
Out[29]:
```



```
In [30]: Image('AvgRank_MainF.png')
```

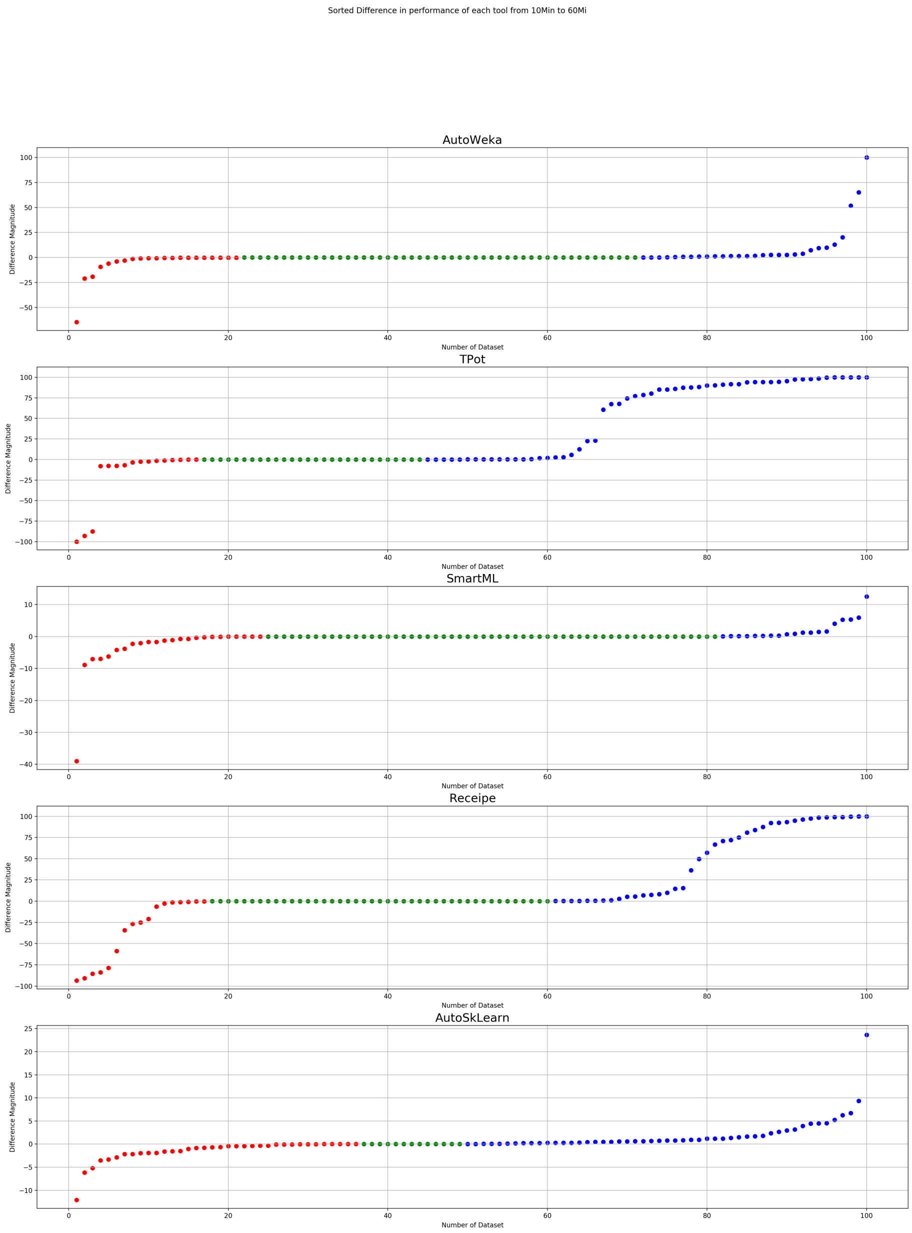
Out[30]:



0.14.1 Performance Gain by increasing time budget

```
In [31]: Image('Diff_MainF_10_60.png')
```

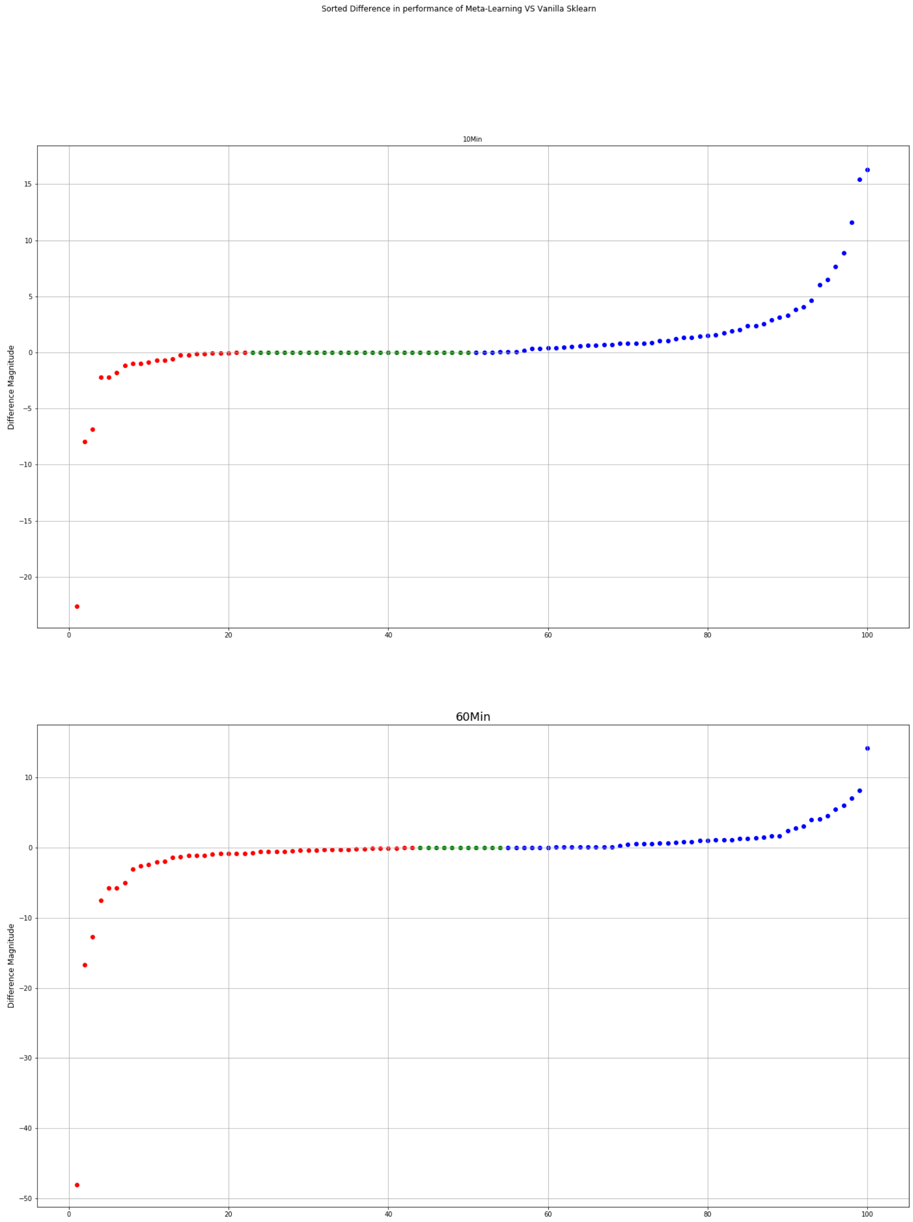
Out[31]:



Meta-Learning Vs Vanilla Version - FScore

```
In [32]: Image('Diff_MetaLearningF.png')
```

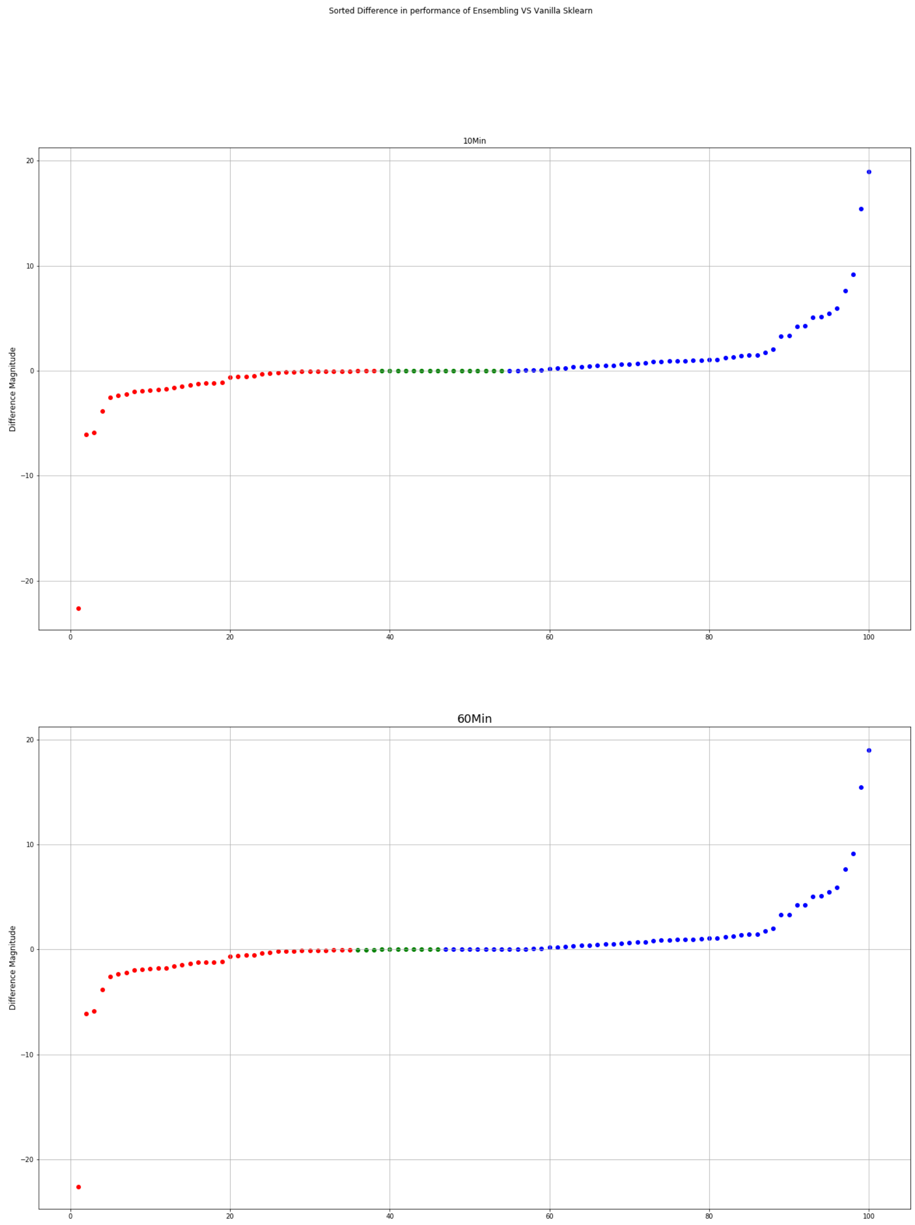
```
Out[32]:
```



Same for ensembling

```
In [33]: Image('Diff_EnsemblingF.png')
```

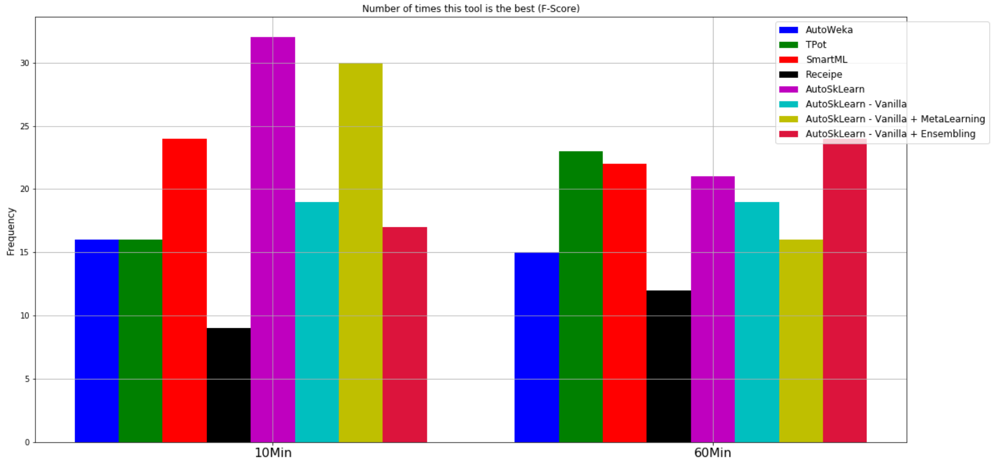
```
Out[33]:
```



How many times each tool has been the best performance?

```
In [34]: Image('Best_AllF.png')
```

```
Out[34]:
```


Number of times this tool is the best (F-Score)
Frequency
0
5
10
15
20
25
30
10Min
60Min
AutoWeka
TPot
SmartML
Receipe
AutoSkLearn
AutoSkLearn - Vanilla
AutoSkLearn - Vanilla + MetaLearning
AutoSkLearn - Vanilla + Ensembling