HAMMEROFLIGHT

__1.7.4__

INSTRUCTIONS FOR INSTALLATION FOR FIRST TIME:

Any one of the following methods can be chosen. The most preferred way is direct installation from the Jupyter Notebook. (Note that the '!` must be used before pip without spaces as shown.)

1. **In command prompt**: pip install hammeroflight
2. **In Jupyter Notebook:** !pip install hammeroflight
3. **In Anaconda Powershell prompt:** pip install hammeroflight

INSTRUCTIONS FOR INSTALLING AND UPGRADE:

1. **In command prompt**: pip install hammeroflight==x.x.x (Version Number)
2. **In Jupyter Notebook:** !pip install hammeroflight==x.x.x
3. **In Anaconda Powershell prompt:** pip install hammeroflight==x.x.x

**The latest Version Number is written in the Header Section, or can be found in the README.txt

** Follow Github to know version number.

DEMO OF SOME AVAILABLE FUNCTIONS:

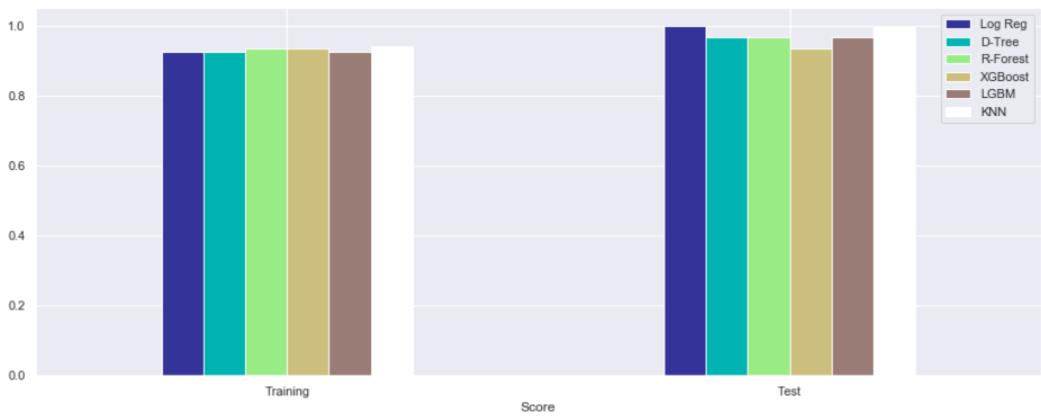
HAMMEROFLIGHT.MODELCOMPARATOR

CLF_COMPARATOR( )

In [8]:
```
from hammeroflight.modelcomparator import clf_comparator
clf_comparator(X_train, X_test, y_train, y_test, 10)
```

Out[8]:

Score	Log Reg	D-Tree	R-Forest	XGBoost	LGBM	KNN
Training	0.925	0.925000	0.933333	0.933333	0.925000	0.941667
Test	1.000	0.966667	0.966667	0.933333	0.966667	1.000000



HAMMEROFLIGHT

__1.7.4__

HAMMEROFLIGHT.MODELCOMPARATOR

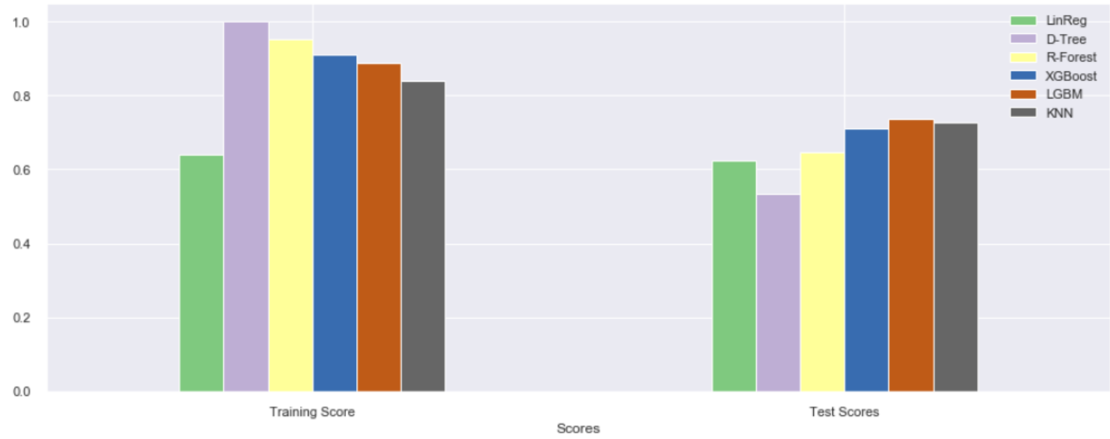
REG_COMPARATOR( )

```
from hammeroflight.modelcomparator import reg_comparator
from hammeroflight.modelfitter import fit_regress
```

```
reg_comparator(X_train, X_test, y_train, y_test)
```

```
[12:00:11] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/objective/regression_obj.cu:152: reg:linear is now depr
ecated in favor of reg:squarederror.
```

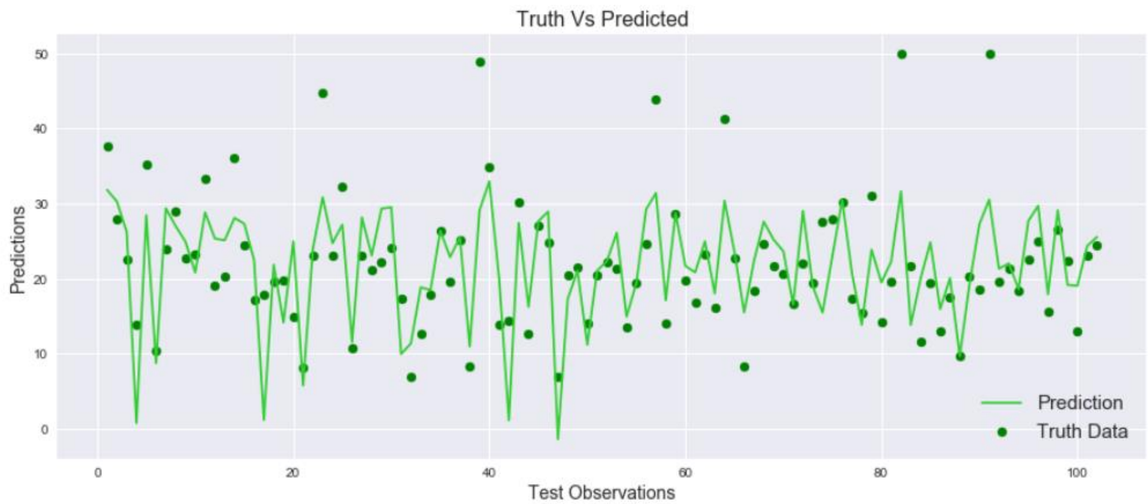
Scores	LinReg	D-Tree	R-Forest	XGBoost	LGBM	KNN
Training Score	0.641401	1.000000	0.953221	0.910335	0.888643	0.839217
Test Scores	0.625071	0.534098	0.647935	0.710684	0.737306	0.725607
RMSE	5.765800	6.427300	5.587200	5.064900	4.826200	4.932500



HAMMEROFLIGHT.PLOTTER

TESTPLOT( )

```
from hammeroflight.plotter import testplot
testplot(y_test, y_pred)
```



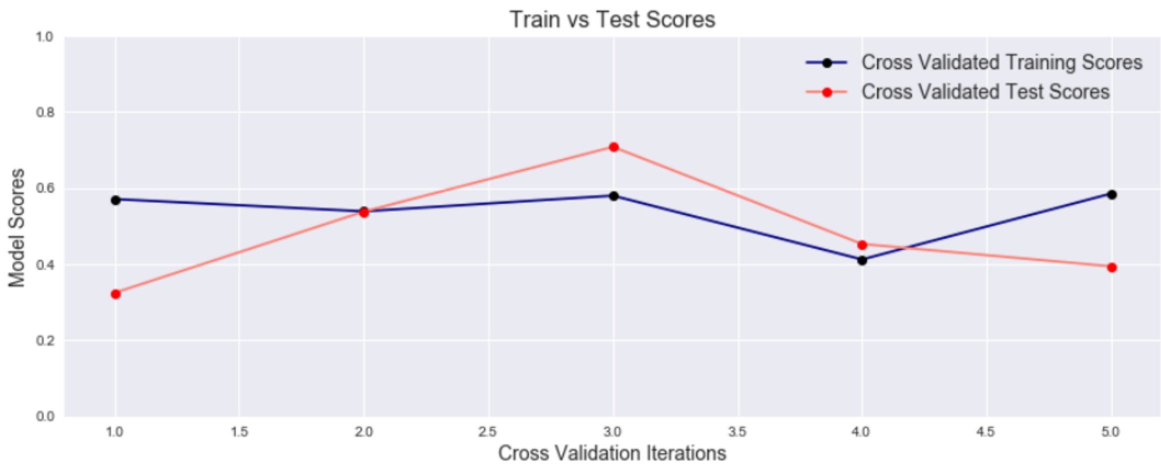
HAMMEROFLIGHT.MODELFITTER

RUN_REGRESSOR( )

```
from lightgbm import LGBMRegressor
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lg = LGBMRegressor(learning_rate=0.01)
run_regressor(lr, X_train, X_test, y_train, y_test, 5)
```

```
Predictions stored in global variable "pred".
```

	Score
CV Training Score	53.7573
CV Test Score	48.3721
RMSE	6.23072
MAE	4.5521
MAPE %	22.7647
Fit	Over-Fitted



HAMMEROFLIGHT

__1.7.4__

HAMMEROFLIGHT.MODELFITTER

RUN_CLASSIFIER( )

In [22]:
```
from hammeroflight.modelfitter import run_classifier
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=20)
run_classifier(rf, X_train, X_test, y_train, y_test, 10)
```

```
Predictions stored in global variable "pred".
```

Out[22]:

	Score
CV Training Score	89.5842
CV Test Score	89.4153
Precision	0.868108
Recall	0.894638
F1-Score	0.249601
Fit	Good Fit

HAMMEROFLIGHT.PLOTTER

FITTINGPLOT( )

```
In [28]: A = df['RM']
         b = df['MEDV']

         A = np.asarray(A).reshape(-1,1)
```

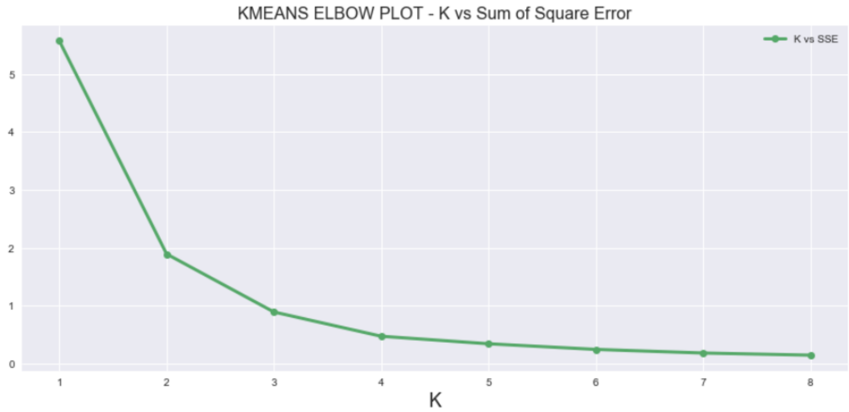
```
In [29]: from hammmeroflight.modelfitter import fittingplot
         fittingplot(lg, A, b)
```

HAMMEROFLIGHT.MODELFITTER

KMEANS_KFINDER( )

```
In [9]: from hammeroflight.modelfitter import kmeans_kfinder
```

```
In [11]: kmeans_kfinder(df1)
```



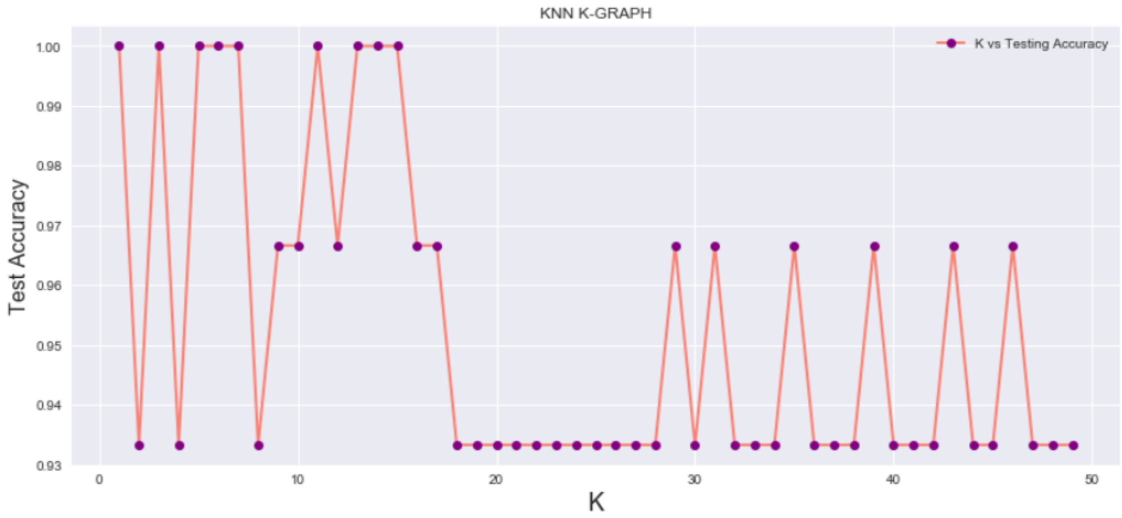
K value seems best at 2 - 4. We will test clustering for all three values.

Elbow Plot to determine best value of K in KMeans Clustering (Unsupervised Learning)

HAMMEROFLIGHT.MODELFITTER

KNN_KFINDER( )

```
from hammeroflight.modelfitter import knn_kfinder
k = knn_kfinder(X_train, X_test, y_train, y_test, 1, 50)
```



Graph to display best Values of K. In this case, K is best between 13 and 17 as further down the test accuracy fluctuates below 0.97.

HAMMEROFLIGHT

__1.7.4__

HAMMEROFLIGHT.ARUFUNCTIONS

QUALITYREPORT( )

Best used as from hammeroflight.arufunctions import qualityreport as qr

```
from hammeroflight.arufunctions import qualityreport, cleanandencode, featureselector
from hammeroflight.modelfitter import fit_classify, fittingplot
from hammeroflight.modelcomparator import clf_comparator
```

```
# Viewing Quality report of the dataset.
qualityreport(df)
```

```
Categorical Features: 9 | Numerical Features: 26 |Dataset Shape: (1470, 35)| DataSet Integrity: 100.0 %
```

	Dtype	Available Rows	Missing Values	Percent Missing	Mean-Mode	Min	Max	No. Of Uniques	Unique Values
Age	int64	1470	0	0.0	35	18	60	43	[41, 49, 37, 33, 27, 32, 59, 30, 38, 36, 35, 2...
Attrition	object	1470	0	0.0	No	No	Yes	2	[Yes, No]
BusinessTravel	object	1470	0	0.0	Travel_Rarely	Non-Travel	Travel_Rarely	3	[Travel_Rarely, Travel_Frequently, Non-Travel]
DailyRate	int64	1470	0	0.0	691	102	1499	886	[1102, 279, 1373, 1392, 591, 1005, 1324, 1358,...
Department	object	1470	0	0.0	Research & Development	Human Resources	Sales	3	[Sales, Research & Development, Human Resources]
DistanceFromHome	int64	1470	0	0.0	2	1	29	29	[1, 8, 2, 3, 24, 23, 27, 16, 15, 26, 19, 21, 5...
Education	int64	1470	0	0.0	3	1	5	5	[2, 1, 4, 3, 5]
EducationField	object	1470	0	0.0	Life Sciences	Human Resources	Technical Degree	6	[Life Sciences, Other, Medical, Marketing, Tec...
EmployeeCount	int64	1470	0	0.0	1	1	1	1	[1]
EmployeeNumber	int64	1470	0	0.0	1	1	2068	1470	[1, 2, 4, 5, 7, 8, 10, 11, 12, 13, 14, 15, 16,...
EnvironmentSatisfaction	int64	1470	0	0.0	3	1	4	4	[2, 3, 4, 1]

HAMMEROFLIGHT.ARUFUNCTIONS

IMPUTE_ENCODE( )

	Emp_ID	Name	Age	Income	Department	Posting
0	P001	Aru	35	11000.0	AI	Tier 1
1	P002	Mahesh	28	6000.0	Sales	Tier 2
2	P003	Ranjit	36	9000.0	ML	NaN
3	P004	Abhishek	34	8700.0	Marketing	Tier 2
4	P005	Supriya	36	13000.0	Top Management	Tier 1

```
from hammeroflight.arufunctions import impute_encode
df = impute_encode(df)
df.head()
```

	Age	Income	Department	Posting
0	35	11000.0	AI	0
1	28	6000.0	Sales	1
2	36	9000.0	ML	0
3	34	8700.0	Marketing	1
4	36	13000.0	Top Management	0

EMP_ID dropped

Posting Label Encoded

Department not touched

Missing Values imputed by mean/mode

HAMMEROFLIGHT

__1.7.4__

With Dummy set to true: All the remaining unencoded variables are transformed to One Hot Encoded, drop_first=True.

IMPUTE_ENCODE (dummy=True**)**

```
df = impute_encode(df, dummy=True)
df.head()
```

	Age	Income	Posting	Department_Assistant	Department_ML	Department_Marketing	Department_Sales	Department_Top Management
0	35	11000.0	0	0	0	0	0	0
1	28	6000.0	1	0	0	0	1	0
2	36	9000.0	0	0	1	0	0	0
3	34	8700.0	1	0	0	1	0	0
4	36	13000.0	0	0	0	0	0	1

```
qr(df)
```

Categorical Features: 0 | Numerical Features: 8 | Dataset Shape: (19, 8) | DataSet Integrity: 100.0 %

	Dtype	Available Rows	Missing Values	Percent Missing	Mean-Mode	Min	Max	No. Of Uniques	Unique Values
Age	int64	19	0	0.0	37.315789	28.0	63.0	13	None
Income	float64	19	0	0.0	9508.333333	3400.0	22000.0	18	None
Posting	int32	19	0	0.0	0.736842	0.0	2.0	3	None
Department_Assistant	uint8	19	0	0.0	0.157895	0.0	1.0	2	None
Department_ML	uint8	19	0	0.0	0.263158	0.0	1.0	2	None
Department_Marketing	uint8	19	0	0.0	0.105263	0.0	1.0	2	None
Department_Sales	uint8	19	0	0.0	0.157895	0.0	1.0	2	None
Department_Top Management	uint8	19	0	0.0	0.157895	0.0	1.0	2	None

OTHER USEFUL FUNCTIONS:

1. featureselector( ) – Correlation based Feature Selector for ML algorithms
2. arima_ordertuner( ) – p, d, q values for ARIMA forecasting model hypertuning.
3. plot_forecast( ) – Forecast plotting of Truth and Predicted Values
4. cleanandencode( ) – Similar to Impute_Encode except no imputation of NaNs.

■ Aru Raghuvanshi