Capstone Project 1: In-Depth Analysis

The Italia dataset is for 2 months i.e. Nov and Dec 2013. The data during the last two weeks of december is seasonal (christmas and new year week) it was decided that this data will not be used. Remaining 6 weeks of data will be used as follows

1. Training data - week 1 to week 3
2. Test data - week 4
3. Hold out data - week 5 and week 6

Parameters used to evaluate the models' performance:

Execution time: The time taken by each model to fit the training data and predict the accuracy check score on the testing data.
Score: R^2 score calculated based on the test data against the predicted and original values.

2 Different Approaches for prediction:

Since we have time series data one approach to generate predictions is using the **Autoregressive models** (we have seen in EDA that the data is auto regressive with 24 hours lag). The other approach is to generate additional variables t-1 and t-2 data i.e. take the **correlated variable** t's one hour and two hour lag data as features to derive the dependent variable (we have seen in EDA that some variables are highly correlated)

Correlated variable model:

As described above 2 additional variables were created for each independent variable.

```
hourlydf['callIn-1'] = hourlydf.groupby('cellId')['callIn'].apply(lambda x: x.shift(1))
hourlydf['callIn-2'] = hourlydf.groupby('cellId')['callIn'].apply(lambda x: x.shift(2))
hourlydf['callOut-1'] = hourlydf.groupby('cellId')['callOut'].apply(lambda x: x.shift(1))
hourlydf['callOut-2'] = hourlydf.groupby('cellId')['callOut'].apply(lambda x: x.shift(2))
hourlydf['SMSin-1'] = hourlydf.groupby('cellId')['SMSin'].apply(lambda x: x.shift(1))
hourlydf['SMSin-2'] = hourlydf.groupby('cellId')['SMSin'].apply(lambda x: x.shift(2))
```

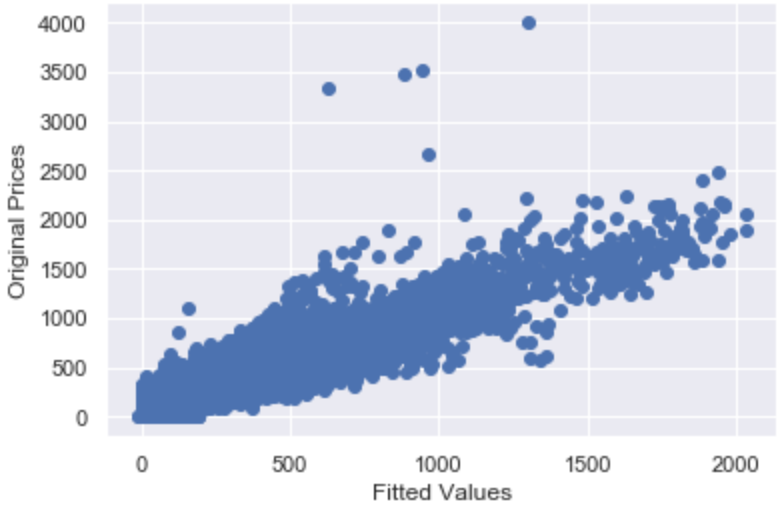
The data is split into Training, test and hold out data as described above

```
train_data = hourlydf[hourlydf.index< '23-11-2013']
test_data = hourlydf[(hourlydf.index> '23-11-2013') & (hourlydf.index< '30-11-2013')]
hold_data = hourlydf[(hourlydf.index> '30-11-2013') & (hourlydf.index< '14-12-2013')]
```

Different modelling techniques were used to generate the SMSin prediction.

Linear Regression:

OLS:



L2 regularization (ridge regression):

No difference in performance for a variety of alpha values

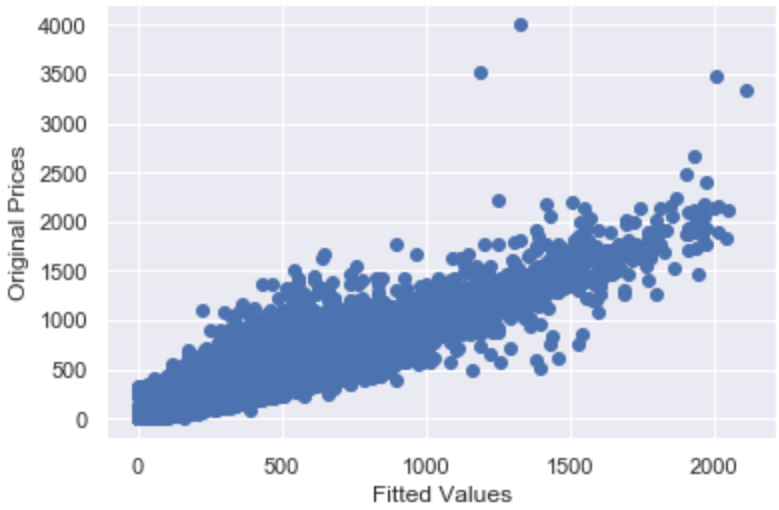
```
alpha: -200.0, score 0.9275185146620516
alpha: -178.94736842105263, score 0.9275185146812248
alpha: -157.89473684210526, score 0.9275185147003979
alpha: -136.8421052631579, score 0.9275185147195708
alpha: -115.78947368421052, score 0.9275185147387435
alpha: -94.73684210526315, score 0.9275185147579158
alpha: -73.68421052631578, score 0.927518514777088
alpha: -52.63157894736841, score 0.9275185147962601
alpha: -31.57894736842104, score 0.927518514815432
alpha: -10.52631578947367, score 0.9275185148346036
alpha: 10.5263157894737, score 0.927518514853775
alpha: 31.57894736842107, score 0.9275185148729461
alpha: 52.63157894736844, score 0.9275185148921172
alpha: 73.68421052631584, score 0.9275185149112881
alpha: 94.73684210526318, score 0.9275185149304587
alpha: 115.78947368421052, score 0.9275185149496291
alpha: 136.84210526315792, score 0.9275185149687993
alpha: 157.89473684210532, score 0.9275185149879693
alpha: 178.94736842105266, score 0.9275185150071391
alpha: 200.0, score 0.9275185150263088
```

KNearest Neighbors:

A subset of the training data was taken to optimize the hyperparameter K. It was observed the best fit would be for 10 neighbors.

```
k: 1, mean: 0.8283656104853211
k: 2, mean: 0.8586630553049225
k: 3, mean: 0.8688096229999748
k: 4, mean: 0.8737146045225771
k: 5, mean: 0.876097119383212
k: 6, mean: 0.877168613992035
k: 7, mean: 0.8790661772377913
k: 8, mean: 0.8791225000540396
k: 9, mean: 0.8791414326169289
k: 10, mean: 0.8792743387749277
k: 11, mean: 0.8789702191591946
k: 12, mean: 0.8790995135229762
k: 13, mean: 0.8790249069787825
k: 14, mean: 0.8788505206949879
k: 15, mean: 0.8785756507248701
k: 16, mean: 0.8782781927783727
k: 17, mean: 0.8779191951179642
k: 18, mean: 0.8775313986019717
k: 19, mean: 0.8771809242204373
k: 20, mean: 0.8767290960986487
k: 21, mean: 0.8764238061607527
k: 22, mean: 0.8763336824738637
k: 23, mean: 0.8758617423194751
k: 24, mean: 0.8755569385214548
k: 25, mean: 0.8752467263113233
k: 26, mean: 0.8746678286090244
k: 27, mean: 0.8739683172305268
k: 28, mean: 0.8732868891487963
k: 29, mean: 0.8728602926647171
k: 30, mean: 0.8721946815621354
k: 31, mean: 0.8713753061950368
k: 32, mean: 0.8708392181875844
k: 33, mean: 0.8699860989865206
k: 34, mean: 0.8691817102415215
```

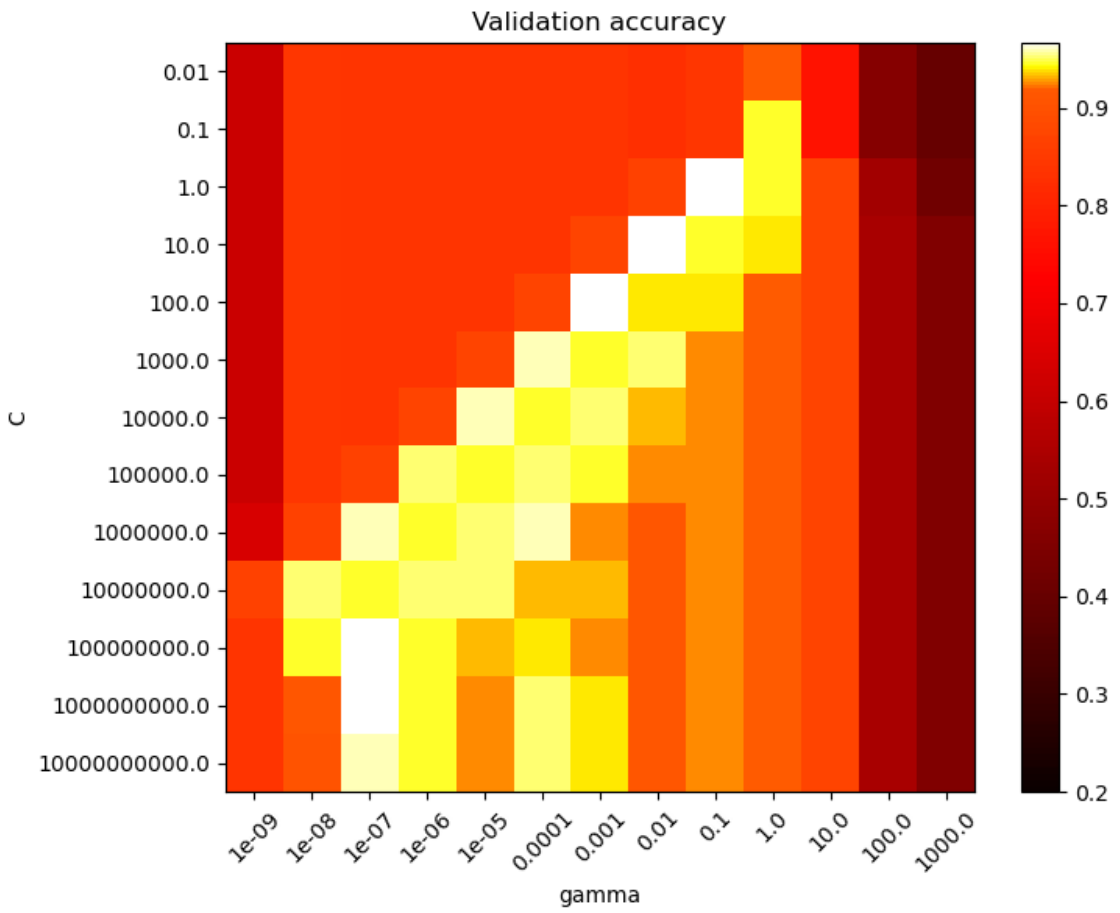
Parameters:
Neighbors = 5 (default), Score = 93.4%
Parameters:
Neighbors = 10 (optimization), Score = 94%



SVM:

Referring to this resource I decided to take C=1 and gamma = 0.1 as the hyperparameters (any higher value of C was taking very long time to process)

Parameters:
C = 1, gamma = 0.1, Score: 87%



Random Forest:

GridSearchCV was applied to identify the best hyperparameters for Random Forest model, best model is given below:

Parameters:
'bootstrap': True,
'max_depth': 30,
'max_features': 'auto',
'min_samples_leaf': 4,
'min_samples_split': 5,
'n_estimators': 600

```
Fitting 3 folds for each of 4320 candidates, totalling 12960 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  33 tasks      | elapsed:   16.7s
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed:  1.3min
[Parallel(n_jobs=-1)]: Done 357 tasks      | elapsed:  2.8min
[Parallel(n_jobs=-1)]: Done 640 tasks      | elapsed:  4.8min
[Parallel(n_jobs=-1)]: Done 1005 tasks      | elapsed:  7.8min
[Parallel(n_jobs=-1)]: Done 1450 tasks      | elapsed: 11.5min
[Parallel(n_jobs=-1)]: Done 1977 tasks      | elapsed: 15.7min
[Parallel(n_jobs=-1)]: Done 2584 tasks      | elapsed: 20.2min
[Parallel(n_jobs=-1)]: Done 3273 tasks      | elapsed: 25.5min
[Parallel(n_jobs=-1)]: Done 4042 tasks      | elapsed: 32.4min
[Parallel(n_jobs=-1)]: Done 4893 tasks      | elapsed: 38.9min
[Parallel(n_jobs=-1)]: Done 5824 tasks      | elapsed: 46.3min
[Parallel(n_jobs=-1)]: Done 6837 tasks      | elapsed: 54.6min
[Parallel(n_jobs=-1)]: Done 7930 tasks      | elapsed: 65.1min
[Parallel(n_jobs=-1)]: Done 9105 tasks      | elapsed: 75.6min
[Parallel(n_jobs=-1)]: Done 10360 tasks      | elapsed: 87.0min
[Parallel(n_jobs=-1)]: Done 11697 tasks      | elapsed: 99.6min
[Parallel(n_jobs=-1)]: Done 12960 out of 12960 | elapsed: 110.7min finished
```

```
Wall time: 1h 50min 40s
```

```
{'bootstrap': True,
 'max_depth': 30,
 'max_features': 'auto',
 'min_samples_leaf': 4,
 'min_samples_split': 5,
 'n_estimators': 600}
```

Autoregressive Models:

ARIMA:

A 24 Hour lag was applied on the ARIMA model with 0 trend and displacement.

ARMA Model Results

Dep. Variable:	SMSin	No. Observations:	144
Model:	ARMA(24, 0)	Log Likelihood	-835.355
Method:	css-mle	S.D. of innovations	1555.894
Date:	Tue, 18 Aug 2020	AIC	1720.709
Time:	09:51:32	BIC	1794.955
Sample:	11-01-2013 - 11-06-2013	HQIC	1750.878

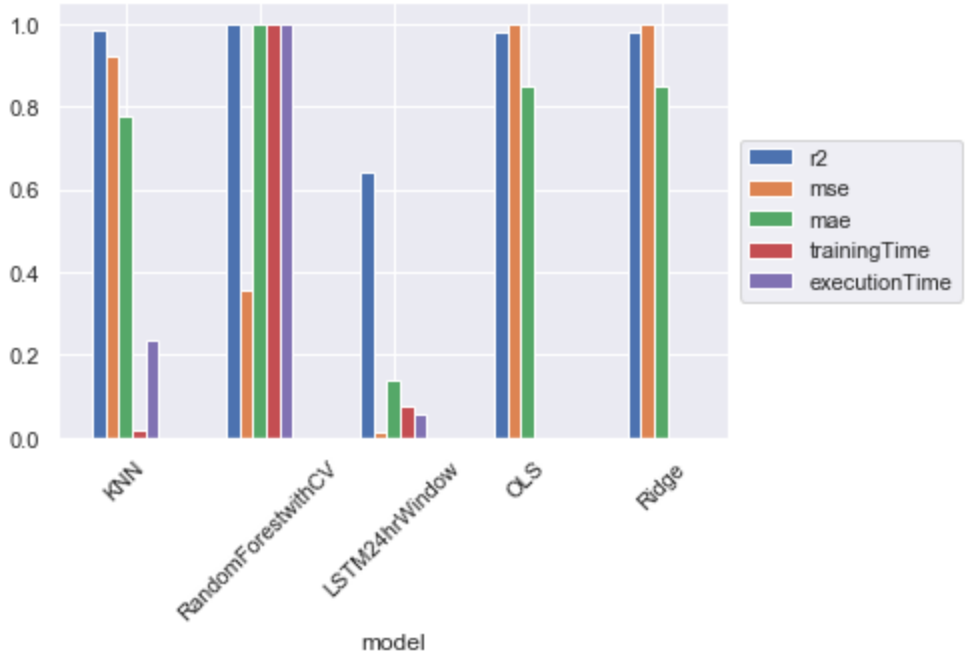
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1.SMSin	0.8709	5.25e-06	1.66e+05	0.000	0.871	0.871
ar.L2.SMSin	0.2222	nan	nan	nan	nan	nan
ar.L3.SMSin	-0.4176	nan	nan	nan	nan	nan
ar.L4.SMSin	0.0662	nan	nan	nan	nan	nan
ar.L5.SMSin	0.2517	4.94e-06	5.09e+04	0.000	0.252	0.252
ar.L6.SMSin	-0.1926	4.94e-06	-3.9e+04	0.000	-0.193	-0.193
ar.L7.SMSin	-0.1171	3.33e-07	-3.51e+05	0.000	-0.117	-0.117
ar.L8.SMSin	0.1523	nan	nan	nan	nan	nan
ar.L9.SMSin	-0.0668	nan	nan	nan	nan	nan
ar.L10.SMSin	-0.0192	nan	nan	nan	nan	nan
ar.L11.SMSin	0.0942	7.17e-06	1.31e+04	0.000	0.094	0.094
ar.L12.SMSin	-0.0459	6.57e-06	-6982.105	0.000	-0.046	-0.046
ar.L13.SMSin	-0.0253	7.47e-06	-3383.420	0.000	-0.025	-0.025
ar.L14.SMSin	0.0215	1.35e-05	1591.485	0.000	0.022	0.022
ar.L15.SMSin	-0.0036	1.3e-05	-279.000	0.000	-0.004	-0.004
ar.L16.SMSin	-0.0318	1.04e-05	-3070.747	0.000	-0.032	-0.032
ar.L17.SMSin	-0.0013	4.53e-06	-287.453	0.000	-0.001	-0.001
ar.L18.SMSin	0.0043	nan	nan	nan	nan	nan
ar.L19.SMSin	0.2074	5e-07	4.15e+05	0.000	0.207	0.207
ar.L20.SMSin	-0.2445	nan	nan	nan	nan	nan
ar.L21.SMSin	-0.0116	4.77e-07	-2.44e+04	0.000	-0.012	-0.012
ar.L22.SMSin	0.3284	nan	nan	nan	nan	nan
ar.L23.SMSin	0.0636	nan	nan	nan	nan	nan
ar.L24.SMSin	-0.1053	nan	nan	nan	nan	nan

LSTM:

An LSTM model with the below parameters was trained on the data for each cell (since consolidated data will lose the time series trend information)

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 24, 128)           67584
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 24, 128)           0
_________________________________________________________________
lstm_4 (LSTM)                (None, 24, 128)           131584
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 24, 128)           0
_________________________________________________________________
dropout_2 (Dropout)          (None, 24, 128)           0
_________________________________________________________________
lstm_5 (LSTM)                (None, 64)                49408
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 248,641
Trainable params: 248,641
Non-trainable params: 0
_________________________________________________________________
```

Conclusion:



	model	mae	mse	r2	trainingTime	executionTime
0	KNN	6.34	327.27	0.9333	149.00	42.90
0	RandomForestwithCV	8.16	127.44	0.9477	7800.00	180.00
0	LSTM24hrWindow	1.15	5.19	0.6078	600.00	10.00
0	OLS	6.91	355.92	0.9300	3.18	0.20
0	Ridge	6.91	355.92	0.9300	0.37	0.15

Of all the different models implemented it was observed that the Random Forest model shows the best R^2 score of 94.7% for the selected hyperparameters. Even though the accuracy of the model is very high, the result set for the given hyperparameters took **130** mins to produce this result, in contrast KNN (12) produced a R^2 score of 94% taking **12** mins which is much better than RandomForest. Linear Regression with Ridge regularization was very fast (300 ms) with an R^2 score of 93%.

Depending on the production application i.e. if the model training time is not relevant then RandomForest can be utilised.

For our scenario of predicting the network usage can be done beforehand and we don't require real time feedback. For this reason I have chosen KNN as the correct model since it is not taking very long to execute and also not sacrificing too much on the predictive power.