Predictive Maintenance

DATA SCIENCE PROJECT MILESTONE REPORT

Problem

Failure prediction is a major topic in predictive maintenance in many industries. Airlines are particularly interested in predicting equipment failures in advance so that they can enhance operations, cut cost of time-based preventive maintenance, and reduce flight delays.

Observing engine's health and condition through sensors and telemetry data is assumed to facilitate this type of maintenance by predicting Time-To-Failure (TTF) or Remaining Useful Life (RUL) of in-service equipment. The project is trying to answer the following question:

By using aircraft engine's sensors measurements, can we predict engine's TTF?

Client

A hypothetical airline company in a case-study provided by Microsoft Cortana Intelligence platform.

Approach

By exploring aircraft engine's sensor values over time, machine learning algorithm can learn the relationship between sensor values and changes in sensor values to the historical failures in order to predict failures in the future.

Supervised machine learning algorithms will be used to make the following predictions:

- Use of Regression algorithms to predict engine TTF
- Use of Binary Classification algorithms to predict if the engine will fail in this period
- Use of Multiclass Classification algorithms to predict the period an engine will fail

Data Sets

Files contain simulate aircraft engine run-to-failure events, operational settings, and sensors measurements were provided by Microsoft Cortana Intelligence.

- Training data file contains aircraft engines' run-to-failure data. 20,000+ cycle records for 100 engines.
- Test data file contains aircraft engines' operating data without failure events recorded. Remaining cycles for each engine is provided in a separate Ground truth data file.
- Ground truth data file contains the true remaining cycles for each engine in the test data.

Sample training data:

id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	s6	s7	s8	...	s15	s16	s17	s18	s19	s20	s21
1	1	-0.0007	-0.0004	100	518.67	641.82	1589.7	1400.6	14.62	21.61	554.36	2388.06		8.4195	0.03	392	2388	100	39.06	23.419
1	2	0.0019	-0.0003	100	518.67	642.15	1591.82	1403.14	14.62	21.61	553.75	2388.04		8.4318	0.03	392	2388	100	39	23.4236
1	3	-0.0043	0.0003	100	518.67	642.35	1587.99	1404.2	14.62	21.61	554.26	2388.08		8.4178	0.03	390	2388	100	38.95	23.3442
100	198	0.0004	0	100	518.67	643.42	1602.46	1428.18	14.62	21.61	550.94	2388.24		8.5646	0.03	398	2388	100	38.44	22.9333
100	199	-0.0011	0.0003	100	518.67	643.23	1605.26	1426.53	14.62	21.61	550.68	2388.25		8.5389	0.03	395	2388	100	38.29	23.064
100	200	-0.0032	-0.0005	100	518.67	643.85	1600.38	1432.14	14.62	21.61	550.79	2388.26		8.5036	0.03	396	2388	100	38.37	23.0522

- **id**: is the engine ID, ranging from 1 to 100
- **cycle**: per engine sequence, starts from1 to the cycle number where failure had happened
- **setting1** to **setting3**: engine operational settings
- **s1** to **s21**: sensors measurements

Data Wrangling

Source Data:

Display and examine few lines from the top and the bottom of the source data files to prepare for loading the data into Pandas dataframes.

This will be helpful in deciding how to load the data, for example if there are column headers, comments lines at the start or end of the file, empty rows or columns etc. It will also help in identifying field's separator and potential data types.

Load Data:

Taking the above findings into consideration, load the data from source file into Pandas dataframe, assign columns names, and remove empty or unwanted columns.

Run some Pandas dataframe methods to get more information about the loaded data. This includes dataframe .describe(), .head(), .tail(), .dtypes etc. For example, .describe() method will reveal some insights about numeric column data distribution e.g. columns with one constant value that could be excluded from further steps in the pipeline.

Missing Values:

Check for null (NaN) values by running Pandas dataframe method .null().sum(), which gives the total of missing values for each data column. Accordingly, some columns may be excluded from the data if they have high number of missing values.

Alternatively, missing values could be replaced by some statistical measure like mean, median, mode, or some interpolated value based on domain knowledge.

Fortunately, the data set of this project is clean since it mainly contains sensors and machine generated data.

Outliers Detection:

For outlier detection, run Pandas dataframe .quantile() method, SciPy stats.zscore, or manually by identifying the data points above or below [mean ± n standard deviation], where n can take the value 2 or 3 assuming normal distribution.

Pandas dataframe filtering methods could then be used to remove/replace outliers.

Other Data Manipulation:

Join or merge dataframes based on index or common key. For example, labels for test data were in separate source data file. This could be done by using Pandas .concat() and .merge() methods.

Regression and classification labels for training data were created as follows:

- Regression: Time-To-Failure TTF (no. of remaining cycle before failure) for each cycle/engine is the number of cycles between that cycle and the last cycle of the same engine.
- Binary Classification: if the remaining cycles is less than specific number of cycles (e.g. period = 30), the engine will fail in this period, otherwise the engine is fine.
- Multiclass Classification: by segmenting the TTF into cycle bands (e.g. periods: 0-15, 16-30, 30+), we could identify in which period will the engine fail.

For test data, TTF is provided in a separate truth data file. These two files were merged and then classification labels for test data were created the in the same way described above.

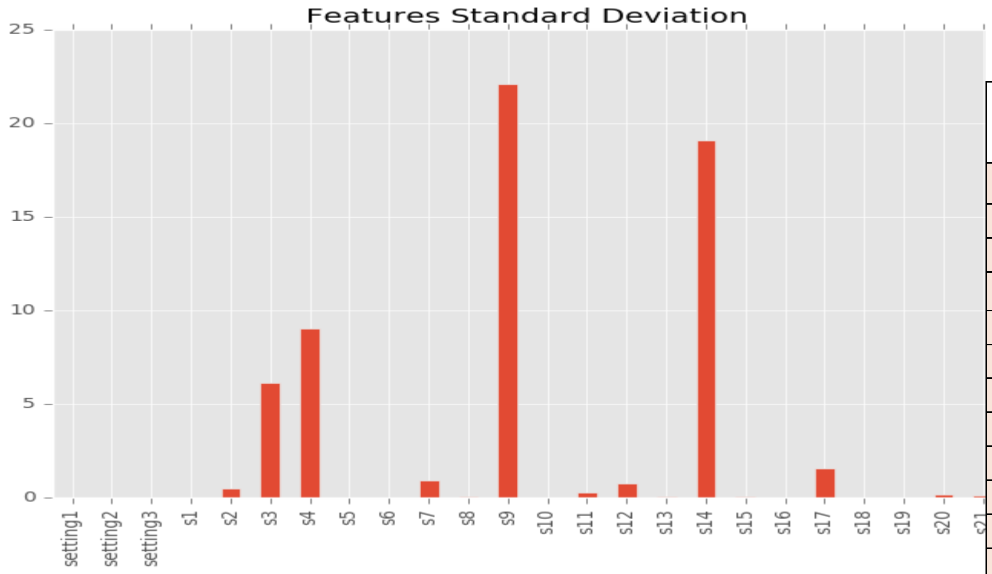
Feature extraction is also applied to the training and test data by introducing additional two columns for each of the 21 sensor columns: rolling mean and rolling standard deviation. This smoothing to the sensors measurements over time would improve the performance of some machine learning algorithms.

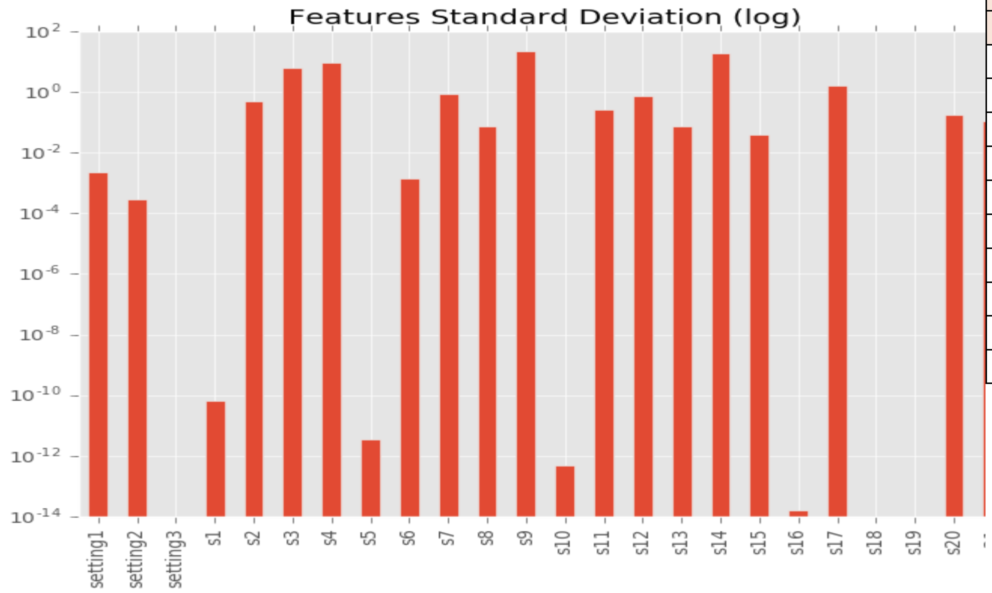
The final training and test data were saved to csv files ready for next steps of analysis and modeling.

Data Wrangling Notebook on GitHub

Findings (EDA)

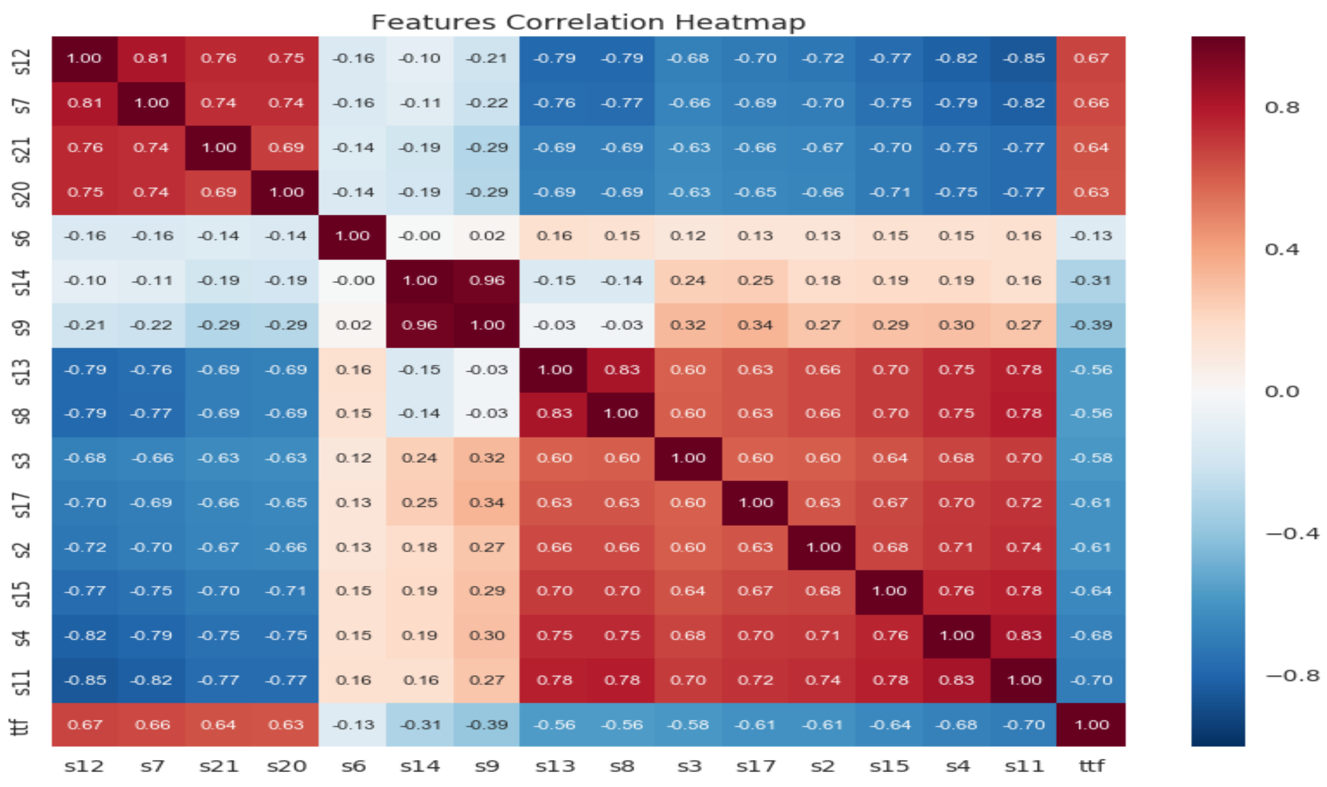
Feature variability, distribution, and correlation were examined to uncover underlying structure and extract important variables.





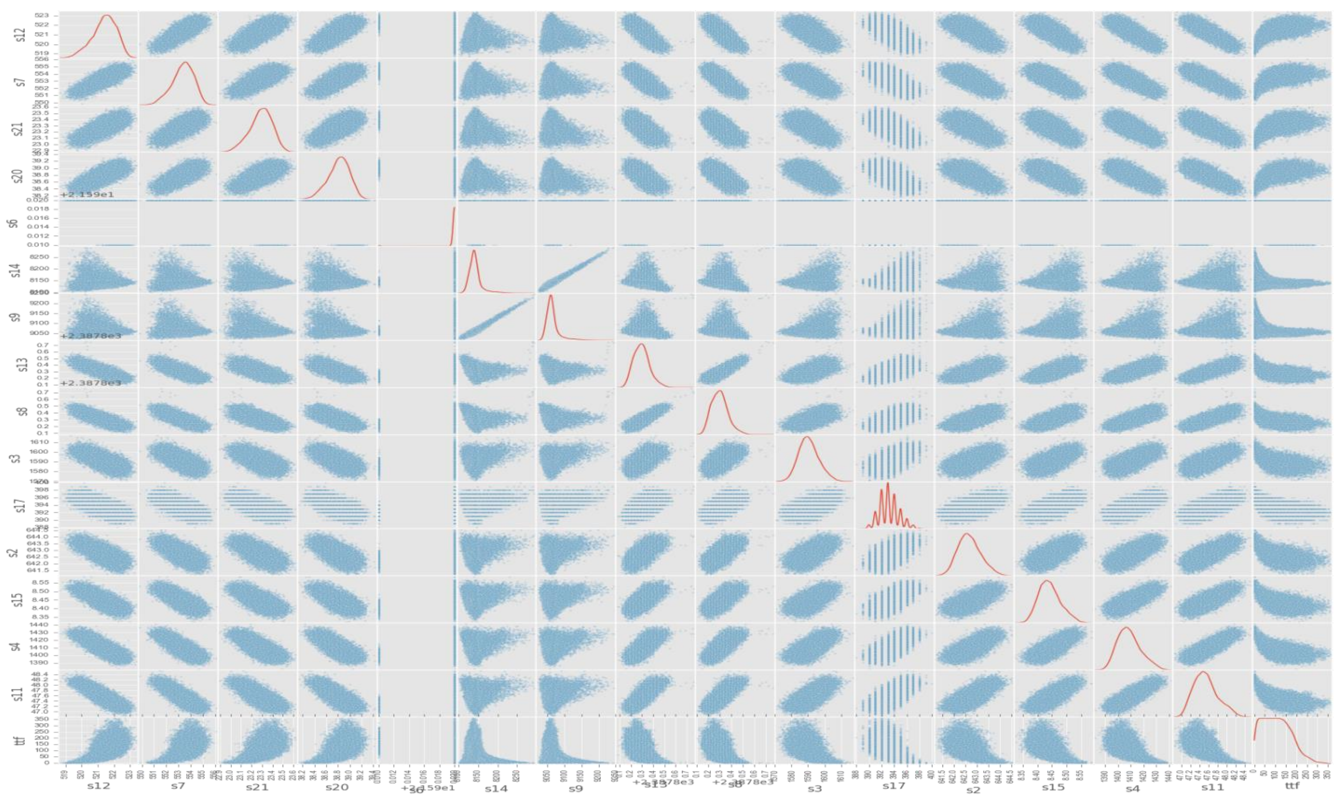
Feature	Standard Deviation
s9	22.0829
s14	19.0762
s4	9.0006
s3	6.1312
s17	1.5488
s7	0.8851
s12	0.7376
s2	0.5001
s11	0.2671
s20	0.1807
s21	0.1083
s13	0.0719
s8	0.0710
s15	0.0375
setting1	0.0022
s6	0.0014
setting2	0.0003
s1	0.0000
s5	0.0000
s10	0.0000
s16	0.0000
s18	-
s19	-
setting3	-

Features with high variability were checked for correlation with other features and regression label (TTF):

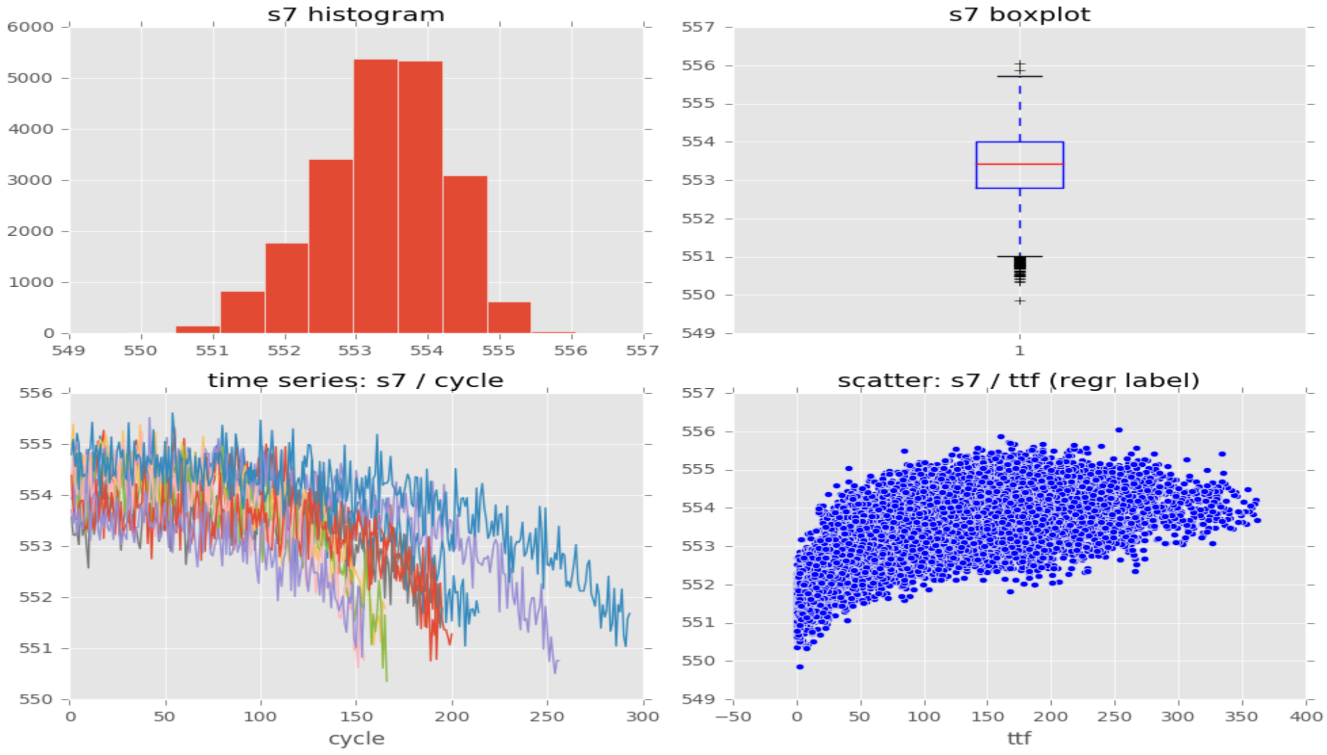


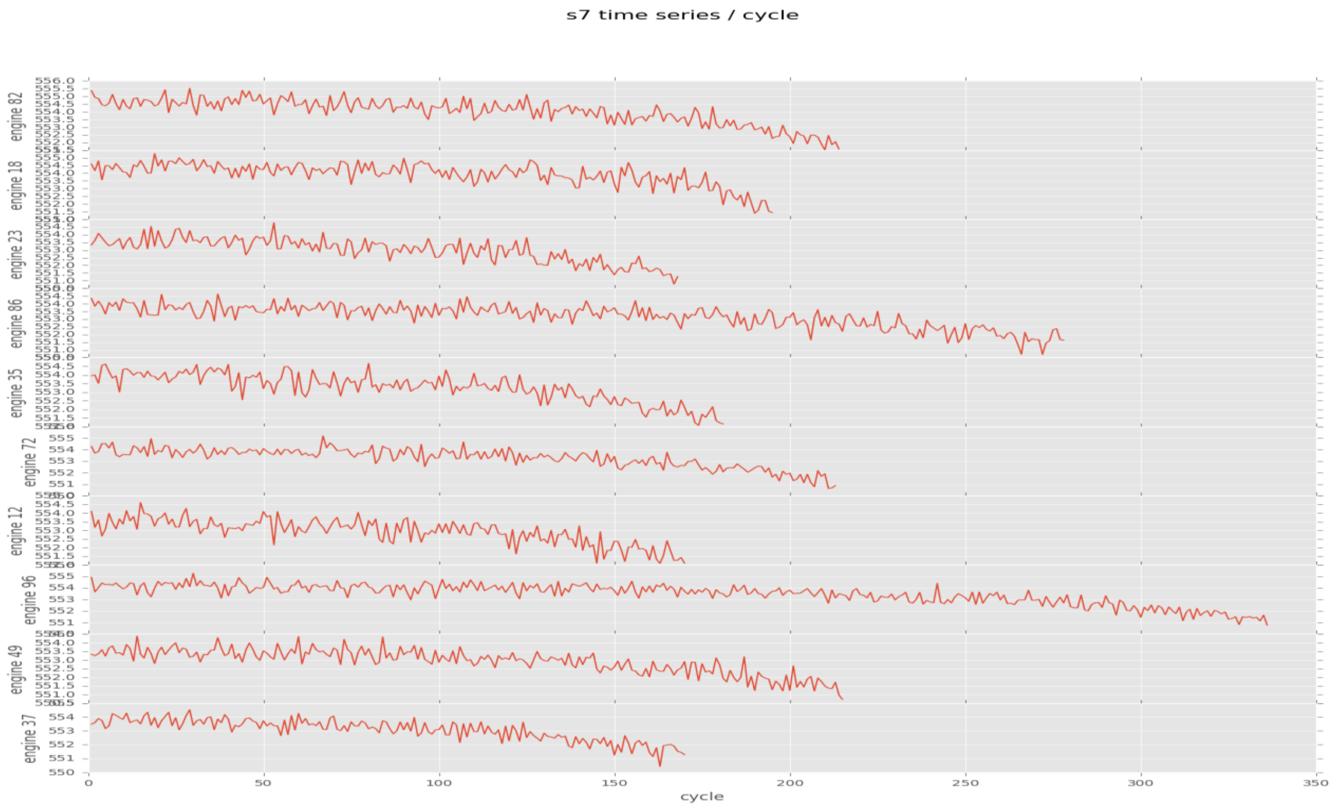
	s12	s7	s21	s20	s6	s14	s9	s13	s8	s3	s17	s2	s15	s4	s11	ttf
s12	1.00	0.81	0.76	0.75	-0.16	-0.10	-0.21	-0.79	-0.79	-0.68	-0.70	-0.72	-0.77	-0.82	-0.85	0.67
s7	0.81	1.00	0.74	0.74	-0.16	-0.11	-0.22	-0.76	-0.77	-0.66	-0.69	-0.70	-0.75	-0.79	-0.82	0.66
s21	0.76	0.74	1.00	0.69	-0.14	-0.19	-0.29	-0.69	-0.69	-0.63	-0.66	-0.67	-0.70	-0.75	-0.77	0.64
s20	0.75	0.74	0.69	1.00	-0.14	-0.19	-0.29	-0.69	-0.69	-0.63	-0.65	-0.66	-0.71	-0.75	-0.77	0.63
s6	-0.16	-0.16	-0.14	-0.14	1.00	-0.00	0.02	0.16	0.15	0.12	0.13	0.13	0.15	0.15	0.16	-0.13
s14	-0.10	-0.11	-0.19	-0.19	-0.00	1.00	0.96	-0.15	-0.14	0.24	0.25	0.18	0.19	0.19	0.16	-0.31
s9	-0.21	-0.22	-0.29	-0.29	0.02	0.96	1.00	-0.03	-0.03	0.32	0.34	0.27	0.29	0.30	0.27	-0.39
s13	-0.79	-0.76	-0.69	-0.69	0.16	-0.15	-0.03	1.00	0.83	0.60	0.63	0.66	0.70	0.75	0.78	-0.56
s8	-0.79	-0.77	-0.69	-0.69	0.15	-0.14	-0.03	0.83	1.00	0.60	0.63	0.66	0.70	0.75	0.78	-0.56
s3	-0.68	-0.66	-0.63	-0.63	0.12	0.24	0.32	0.60	0.60	1.00	0.60	0.60	0.64	0.68	0.70	-0.58
s17	-0.70	-0.69	-0.66	-0.65	0.13	0.25	0.34	0.63	0.63	0.60	1.00	0.63	0.67	0.70	0.72	-0.61
s2	-0.72	-0.70	-0.67	-0.66	0.13	0.18	0.27	0.66	0.66	0.60	0.63	1.00	0.68	0.71	0.74	-0.61
s15	-0.77	-0.75	-0.70	-0.71	0.15	0.19	0.29	0.70	0.70	0.64	0.67	0.68	1.00	0.76	0.78	-0.64
s4	-0.82	-0.79	-0.75	-0.75	0.15	0.19	0.30	0.75	0.75	0.68	0.70	0.71	0.76	1.00	0.83	-0.68
s11	-0.85	-0.82	-0.77	-0.77	0.16	0.16	0.27	0.78	0.78	0.70	0.72	0.74	0.78	0.83	1.00	-0.70
ttf	0.67	0.66	0.64	0.63	-0.13	-0.31	-0.39	-0.56	-0.56	-0.58	-0.61	-0.61	-0.64	-0.68	-0.70	1.00

Scatter matrix was also used to check the distribution and correlation of features:



A number of EDA charts were also used to have more insights on each feature individually, e.g. **S7**:





- There is a very high correlation (> 0.8) between some features e.g.:
  (s14 & s9), (s11 & s4), (s11 & s7), (s11 & s12), (s4 & s12), (s8 & s13), (s7 & s12)
  This multicollinearity may hurt the performance of some machine learning algorithms. So, part of these features will be target for elimination in feature selection during the modeling phase.
- Most features have nonlinear relation with the TTF, hence adding their polynomial transforms may enhance models performance.
- Most features exhibit normal distribution which is likely improves models performance.

Exploratory Data Analysis Notebook on GitHub