1 SUPERMARKET PREDICTION WITH FEATURE TOOL

In [1]:

```
import featuretools as ft
import seaborn as sns
import numpy as np
import pandas as pd

train = pd.read_csv("train.csv")
```

executed in 59.7s, finished 16:26:49 2019-01-29

In [2]:

```
# saving target and features separately
sales = train['Product_Supermarket_Sales']


train.drop(['Product_Supermarket_Sales'], axis=1, inplace=True)
```

executed in 60ms, finished 16:26:49 2019-01-29

In [3]:

```
# imputing missing data
train['Product_Weight'].fillna(method='ffill', inplace = True)
train['Supermarket _Size'].fillna('unknown',inplace=True)
```

executed in 133ms, finished 16:26:49 2019-01-29

In [4]:

```
train.isnull().sum()
```

executed in 229ms, finished 16:26:49 2019-01-29

Out[4]:

```
Product_Identifier               0
Supermarket_Identifier           0
Product_Supermarket_Identifier   0
Product_Weight                   0
Product_Fat_Content              0
Product_Shelf_Visibility         0
Product_Type                     0
Product_Price                    0
Supermarket_Opening_Year         0
Supermarket _Size                0
Supermarket_Location_Type        0
Supermarket_Type                 0
dtype: int64
```

Feature tool is a tool that helps in automatic feature engineering. You can check out the tutorial attached to this notebook folder for some clearer understanding

In [5]:

```
#creating and entity set 'es'
es = ft.EntitySet(id = 'sales')

# adding a dataframe
es.entity_from_dataframe(entity_id = 'bigmart', dataframe = train, index = 'Product_Super
```

executed in 330ms, finished 16:26:50 2019-01-29

Out[5]:

```
Entityset: sales
  Entities:
    bigmart [Rows: 4990, Columns: 12]
  Relationships:
    No relationships
```

In [6]:

```
es.normalize_entity(base_entity_id='bigmart', new_entity_id='outlet', index = 'Supermarke
additional_variables = [ 'Supermarket_Opening_Year', 'Supermarket _Size',
                         'Supermarket_Location_Type', 'Supermarket_Type'])

es.normalize_entity(base_entity_id='bigmart', new_entity_id='product', index = 'Product_I
additional_variables = [ 'Product_Weight','Product_Fat_Content', 'Product_Shelf_Visibilit
                         'Product_Type','Product_Price'])
```

executed in 245ms, finished 16:26:50 2019-01-29

Out[6]:

```
Entityset: sales
  Entities:
    bigmart [Rows: 4990, Columns: 3]
    outlet [Rows: 10, Columns: 5]
    product [Rows: 1451, Columns: 6]
  Relationships:
    bigmart.Supermarket_Identifier -> outlet.Supermarket_Identifier
    bigmart.Product_Identifier -> product.Product_Identifier
```

In [7]:

```
feature_matrix, feature_names = ft.dfs(entityset=es,target_entity='bigmart',
                                       max_depth=4,verbose=1,n_jobs=-1)
```

executed in 20.9s, finished 16:27:11 2019-01-29

```
Built 17 features
EntitySet scattered to workers in 7.524 seconds
Elapsed: 00:06 | Remaining: 00:00 | Progress: 100%|████████████████████████
████████████████| Calculated: 10/10 chunks
```

In [8]:

```
feature_matrix.columns
```

executed in 67ms, finished 16:27:11 2019-01-29

Out[8]:

```
Index(['Product_Identifier', 'Supermarket_Identifier',
       'outlet.Supermarket_Opening_Year', 'outlet.Supermarket _Size',
       'outlet.Supermarket_Location_Type', 'outlet.Supermarket_Type',
       'product.Product_Weight', 'product.Product_Fat_Content',
       'product.Product_Shelf_Visibility', 'product.Product_Type',
       'product.Product_Price', 'outlet.COUNT(bigmart)',
       'outlet.NUM_UNIQUE(bigmart.Product_Identifier)',
       'outlet.MODE(bigmart.Product_Identifier)', 'product.COUNT(bigmart)',
       'product.NUM_UNIQUE(bigmart.Supermarket_Identifier)',
       'product.MODE(bigmart.Supermarket_Identifier)'],
      dtype='object')
```

In [9]:

```
feature_matrix.head()
```

executed in 374ms, finished 16:27:11 2019-01-29

Out[9]:

	Product_Identifier	Supermarket_Identifier	outlet.Supermarket_Ope
Product_Supermarket_Identifier			
DRA12_CHUKWUDI010	DRA12	CHUKWUDI010	
DRA12_CHUKWUDI013	DRA12	CHUKWUDI013	
DRA12_CHUKWUDI017	DRA12	CHUKWUDI017	
DRA12_CHUKWUDI018	DRA12	CHUKWUDI018	
DRA12_CHUKWUDI035	DRA12	CHUKWUDI035	

In [10]:

```
feature_matrix = feature_matrix.reindex(index=train['Product_Supermarket_Identifier'])

feature_matrix=feature_matrix.reset_index()
```

executed in 43ms, finished 16:27:11 2019-01-29

In [11]:

```
feature_matrix.drop(['Product_Supermarket_Identifier','Product_Identifier'],axis=1,inplac
```

executed in 188ms, finished 16:27:12 2019-01-29

In [12]:

```
feature_matrix.nunique()
```

executed in 436ms, finished 16:27:12 2019-01-29

Out[12]:

```
Supermarket_Identifier                          10
outlet.Supermarket_Opening_Year                  9
outlet.Supermarket _Size                         4
outlet.Supermarket_Location_Type                 3
outlet.Supermarket_Type                          4
product.Product_Weight                         388
product.Product_Fat_Content                      3
product.Product_Shelf_Visibility              1373
product.Product_Type                            16
product.Product_Price                         1347
outlet.COUNT(bigmart)                           10
outlet.NUM_UNIQUE(bigmart.Product_Identifier)   10
outlet.MODE(bigmart.Product_Identifier)          3
product.COUNT(bigmart)                           9
product.NUM_UNIQUE(bigmart.Supermarket_Identifier)  9
product.MODE(bigmart.Supermarket_Identifier)    10
dtype: int64
```

In [13]:

```
cat_col=[col for col in feature_matrix if feature_matrix[col].nunique()<=16]
```

executed in 111ms, finished 16:27:12 2019-01-29

In [14]:

```
feature_matrix=pd.get_dummies(feature_matrix,columns=cat_col,prefix=cat_col)
```

executed in 259ms, finished 16:27:12 2019-01-29

Removing values of low variance

In [15]:

```
import sklearn.feature_selection as fs

print(feature_matrix.shape)

## Define the variance threhold and fit the threshold to the feature array.
sel = fs.VarianceThreshold(threshold=(.8 * (1 - .8)))
features_reduced = sel.fit_transform(feature_matrix)

## Print the support and shape for the transformed features
print('Reduced shape:')
print(features_reduced.shape)
```

executed in 15.1s, finished 16:27:28 2019-01-29

```
(4990, 103)
Reduced shape:
(4990, 18)
```

In [16]:

```
print(feature_matrix.columns[sel.get_support()])
```

executed in 34ms, finished 16:27:28 2019-01-29

```
Index(['product.Product_Weight', 'product.Product_Price',
       'outlet.Supermarket _Size_Medium', 'outlet.Supermarket _Size_Small',
       'outlet.Supermarket _Size_unknown',
       'outlet.Supermarket_Location_Type_Cluster 1',
       'outlet.Supermarket_Location_Type_Cluster 2',
       'outlet.Supermarket_Location_Type_Cluster 3',
       'outlet.Supermarket_Type_Supermarket Type1',
       'product.Product_Fat_Content_Low Fat',
       'product.Product_Fat_Content_Normal Fat',
       'outlet.MODE(bigmart.Product_Identifier)_DRA12',
       'outlet.MODE(bigmart.Product_Identifier)_DRA24',
       'product.COUNT(bigmart)_4',
       'product.NUM_UNIQUE(bigmart.Supermarket_Identifier)_4',
       'product.MODE(bigmart.Supermarket_Identifier)_CHUKWUDI010',
       'product.MODE(bigmart.Supermarket_Identifier)_CHUKWUDI013',
       'product.MODE(bigmart.Supermarket_Identifier)_CHUKWUDI017'],
      dtype='object')
```

Removing highly correlated features

In [17]:

```
print(features_reduced.shape)

# Create correlation matrix
corr_matrix = pd.DataFrame(features_reduced,columns=feature_matrix.columns[sel.get_suppor

# Select upper triangle of correlation matrix
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.bool))

# Find index of feature columns with correlation greater than 0.90
to_drop = [column for column in upper.columns if any(upper[column] > 0.90)]

feature_mat=pd.DataFrame(features_reduced,columns=feature_matrix.columns[sel.get_support(

print(to_drop)
print(feature_mat.shape)
```

executed in 1.03s, finished 16:27:29 2019-01-29

```
(4990, 18)
['product.Product_Fat_Content_Normal Fat', 'product.NUM_UNIQUE(bigmart.Super
market_Identifier)_4']
(4990, 16)
```

In [18]:

```
X=feature_mat[:]
Y=sales[:]

Y=Y.ravel()

split_test_size=0.25

from sklearn.model_selection import train_test_split
Xtrain, Xtest, Ytrain, Ytest= train_test_split(X,Y, test_size=split_test_size, random_sta
```

executed in 66ms, finished 16:27:29 2019-01-29

In [19]:

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler().fit(Xtrain)
Xtrain=scaler.transform(Xtrain)
Xtest = scaler.transform(Xtest)
```

executed in 251ms, finished 16:27:29 2019-01-29

In [20]:

```
from catboost import CatBoostRegressor
```

executed in 2.15s, finished 16:27:31 2019-01-29

In [21]:

```
cb=CatBoostRegressor(iterations=500,depth=4,eval_metric='RMSE',
                     random_seed=10,learning_rate=.05,verbose=False)
```

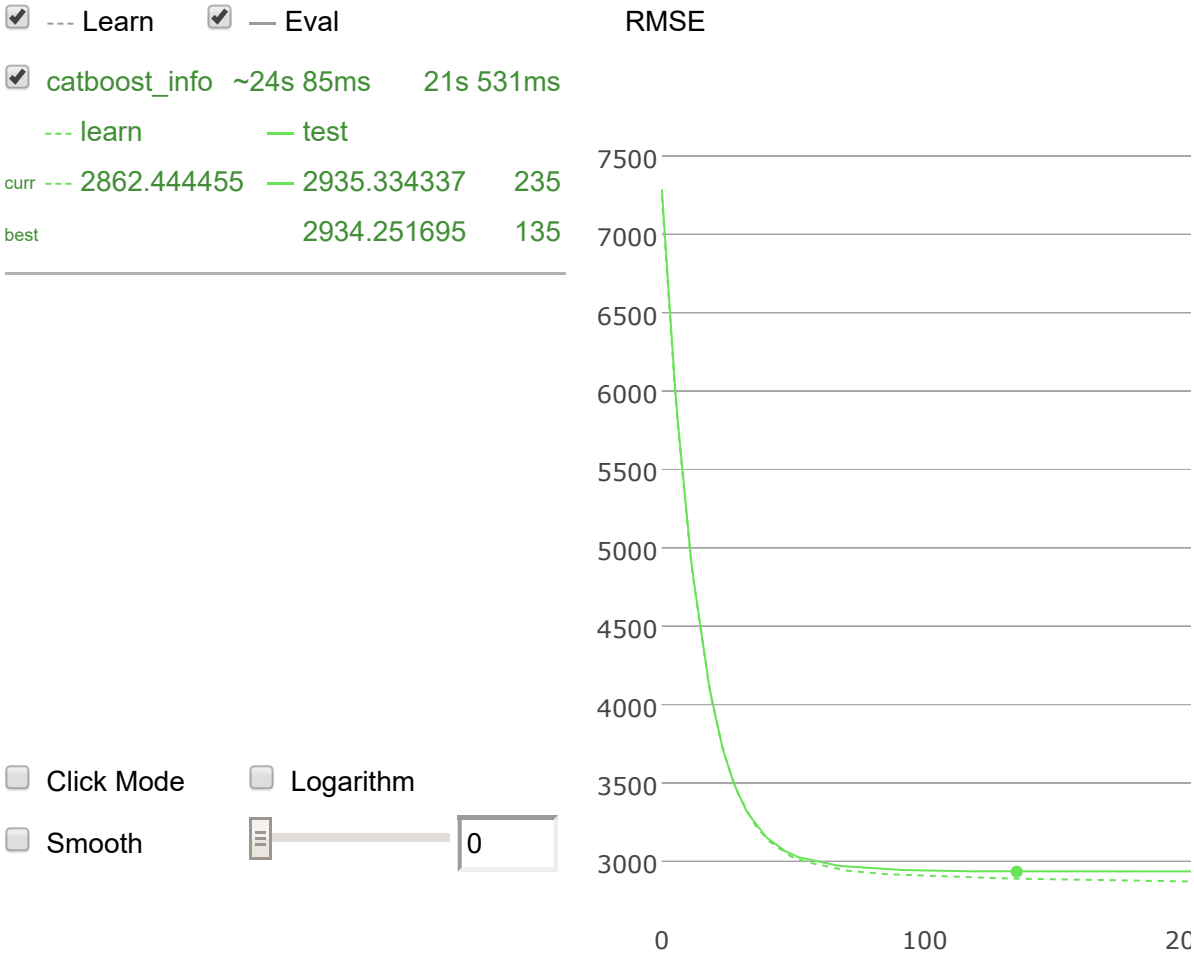
executed in 29ms, finished 16:27:31 2019-01-29

In [22]:

```
cb.fit(Xtrain,Ytrain,use_best_model=True,eval_set=(Xtest,Ytest),plot=True,early_stopping_
```

executed in 23.1s, finished 16:27:54 2019-01-29

```
0:      learn: 7243.8747957     test: 7287.2299530      best: 7287.2299530
(0)     total: 189ms    remaining: 1m 34s
```




```
50:     learn: 3024.1271902     test: 3038.5229253      best: 3038.5229253
(50)    total: 5.34s    remaining: 47s
100:    learn: 2906.0656023     test: 2942.5781720      best: 2942.2436392
(97)    total: 9.95s    remaining: 39.3s
150:    learn: 2882.9302620     test: 2935.1192477      best: 2934.2516950
(135)   total: 14.7s    remaining: 33.9s
200:    learn: 2869.7767033     test: 2934.5984241      best: 2934.2516950
(135)   total: 18.6s    remaining: 27.6s
Stopped by overfitting detector  (100 iterations wait)

bestTest = 2934.251695
bestIteration = 135

Shrink model to first 136 iterations.
```

Out[22]:

```
<catboost.core.CatBoostRegressor at 0x1ab5c1e4d30>
```

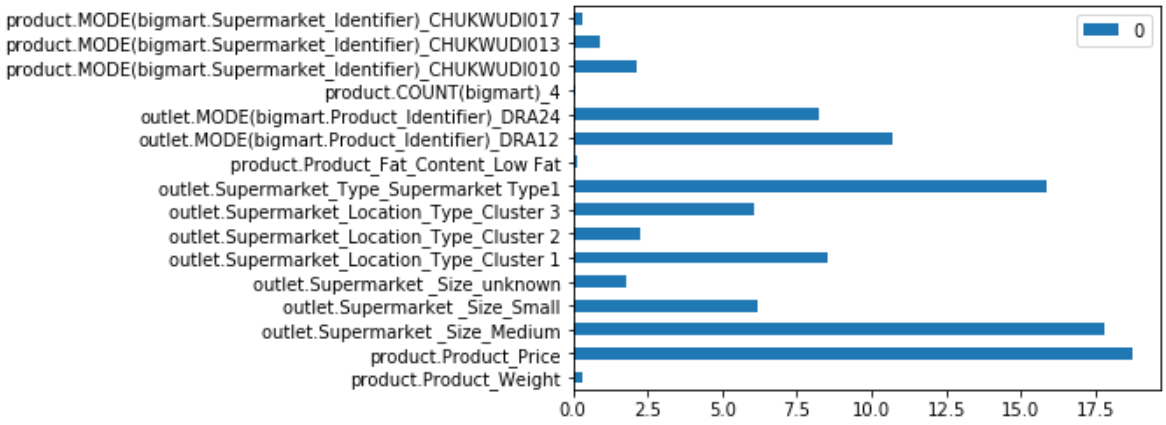
In [23]:

```
b=list(cb.feature_importances_)
pd.DataFrame(index=feature_mat.columns,data=b).plot.barh()
```

executed in 1.96s, finished 16:27:56 2019-01-29

Out[23]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ab5c30bd68>
```



It can be seen that some of our created features from featuretool have high feature_importance showing that the tool is quite useful. This can be particularly useful when we have multiple tables and need to quickly do feature engineering.