Applying machine learning to web lead behavior in order to understand the probability of conversion. FY incremental revenue ~$7 million

# Lead Scoring

Nathan White

1. **Context**

Corporate-wide, Polaris will send nearly [redacted] quote requests to our dealer network across the US, and additionally collect more than [redacted] consumer names through website email or sweepstakes sign-ups. Less than [redacted]% of these will convert to whole goods sales, with ORV leading the way at [redacted]%, on-road and snowmobiles around [redacted]%, followed by Slingshot at [redacted]%. Several customer feedback and market best-practice studies have been conducted to determine the optimal way to increase this conversion rate, and generally the follow-up (or lack thereof) with the consumer who submits the lead correlates with the odds of that consumer completing a purchase. For this reason, several lead follow-up initiatives have been undertaken: outbound calls to leads that provide a phone number, live chat, and automated email lead cultivation, to name a few.

2. **Business Problem**

The marketing team wants to segment leads based on their likelihood to convert, and apply to different marketing tactics to each segment in order serve drive maximum sales for Polaris.

3. **DS 'Project Framing'**

Conduct an analysis to identify which specific traits of a consumer's web session resulting in a lead submittal are most important to converting to sale. Build a live lead-scoring model that will allow marketing to tailor lead cultivation efforts according to the likelihood to close the lead. These learnings will inform the strategy for Polaris corporate lead cultivation, and further enable dealers to best convert leads moving forward.

4. **Technical Considerations – Algorithm development**

   a. **Dependent Variable Definition:**

      Closed lead (yes/no): Whole goods purchase tied to a household within 75 days of a lead submitted (email sign-up, event sign-up, sweepstakes, quote, custom quote, or test ride request).

   b. **Data Sources and Independent Variables:**

      Data sources:

      - CRM
      - Google BigQuery

      Independent variables/attributes:

      - See Appendix A

      Holdout for validation:

      - 50%

   c. **Lead Submission Window:**

      Post implementation of Shift call center (3/1/2016-8/15/2017))

   d. **Purchase Window:**

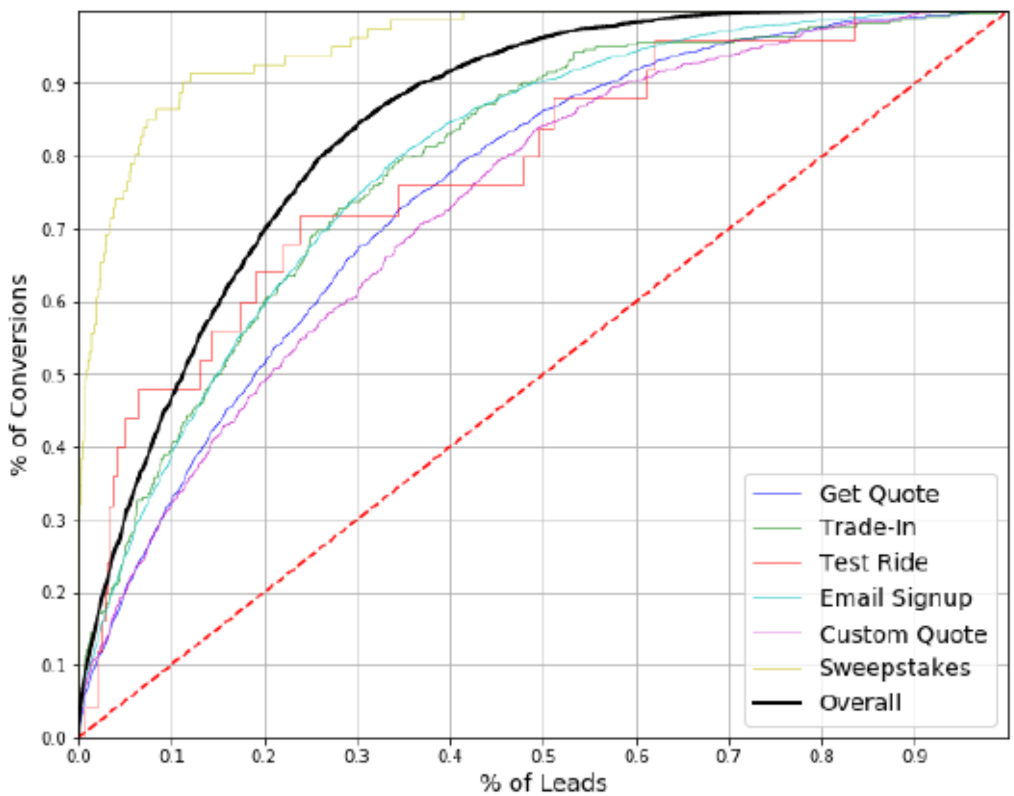
      Any Whole Goods purchase (of any brand) during the 75-day window following lead submission (3/1/2016 – 9/30/2017)

**e. Methodology:**

All independent variables will be assessed via a correlation matrix, then a subset will be selected for modeling using clustering methods and/or random forest models (eliminating highly correlated ones). The scoring model itself will be selected from several various modeling methodologies, evaluated for accuracy and reliability. Logistic regression was used for the final scoring model.

## 5. Results - Model

A high performing model that efficient separates high-from low quality leads. The top 10% of model-scored leads account for over half (55%) of Whole-Goods conversions. The bottom 65% of model scored leads account for less than one-tenth (9%) of Whole-Goods conversions.



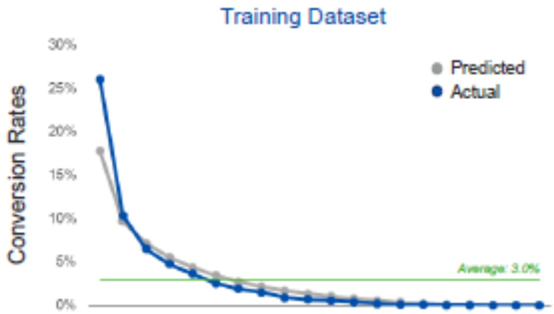

## 6. Results- Business Impact

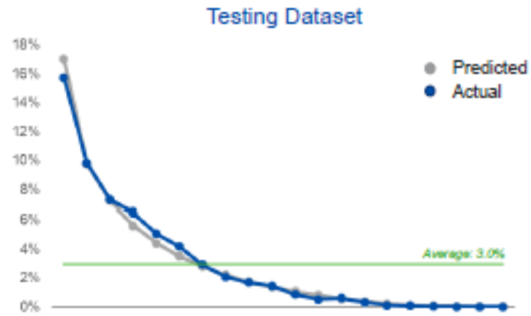
Using the lead-scores, 4 segments of leads were identified and used to create actionable specific marketing action plans depending on their probability to convert.

Sizzling – Leads with the highest probability of conversion. Account for over half (55%) of converted leads. ~40x more likely to convert than cold leads
Hot - Above average leads. Over 14x more likely to convert than Cold leads.
Warm – "Average" leads. ~7x more likely to convert than cold
Cold – Lowest value leads. Over indexed in Sweeps and E-mail sign up lead types.

Based on these segments, marketing spend was shifted within each brand. Less was spent on Cold leads and more on the others. These efforts raised the conversion rate in Hot, Sizzling, and Warm by a range of 0.1% - 0.6%. YTD incremental revenue is $2.3 million, with the expected annual total to top ~$7 million.

## 7. Appendix – Model Validation




**Appendix – Lead Scoring Production Code**

```
import xgboost as xgb
import pandas as pd
import numpy as np
import math
import datetime
from geopy.distance import great_circle
import re
import pickle
from queries import big_queries



#def merge_all_sources():
#     # read in data
#     frames=list()
#
#     for key in big_queries.keys():
#         file_name=big_queries.get(key).get("out_filename")
#         print(file_name)
#         frames.append(pd.read_csv("input_data/" + file_name,
encoding="cp1252"))
#     alldata = pd.concat(frames)
#     #
file_names=["GEMBigQueryCSID_withPrevSessions_appendPurch.csv","SNOBigQuer
yCSID_withPrevSessions_appendPurch.csv"]
#     return alldata

def dev(row):
    if row['devicetype'] == 'desktop':
        rt = 0.028
    elif row['devicetype'] == 'mobile':
        rt = 0.026
    elif row['devicetype'] == 'tablet':
        rt = 0.04
    else:
        rt = 0.029
    return rt
```

```
def brn(row):
    if row['brand'] == 'ACE':
        rt = 0.046
    elif row['brand'] == 'ATV':
        rt = 0.064
    elif row['brand'] == 'GEM':
        rt = 0.01
    elif row['brand'] == 'GRL':
        rt = 0.067
    elif row['brand'] == 'IND':
        rt = 0.011
    elif row['brand'] == 'ORV':
        rt = 0.081
    elif row['brand'] == 'RGR':
        rt = 0.078
    elif row['brand'] == 'RZR':
        rt = 0.049
    elif row['brand'] == 'SLG':
        rt = 0.008
    elif row['brand'] == 'SNO':
        rt = 0.033
    else:
        rt = 0.029
    return rt

def chn(row):
    if row['channel'] == 'Direct':
        rt = 0.022
    elif row['channel'] == 'Display / Banner':
        rt = 0.022
    elif row['channel'] == 'Email':
        rt = 0.018
    elif row['channel'] == 'FB Ads to Web':
        rt = 0.007
    elif row['channel'] == 'Organic':
        rt = 0.046
    elif row['channel'] == 'Paid Search':
        rt = 0.043
    elif row['channel'] == 'Referral':
        rt = 0.005
    elif row['channel'] == 'Remarketing':
        rt = 0.013
    elif row['channel'] == 'Retargeting':
        rt = 0.036
    else:
        rt = 0.029
    return rt

def fnl(row):
    if row['sh_funnelstartdepth']=='1) Phase 1: Brand Exploration (140
Days)':
        rt = 0.028
```

```
    elif row['sh_funnelstartdepth']=='3) Phase 3: Model Details (100
Days)':
        rt = 0.015
    elif row['sh_funnelstartdepth']=='4) Phase 4: Customize and Price (75
Days)':
        rt = 0.036
    elif row['sh_funnelstartdepth']=='6) Quote Form':
        rt = 0.054
    elif row['sh_funnelstartdepth']=='6) Test Ride Form':
        rt = 0.002
    elif row['sh_funnelstartdepth']=='7) Customize Quote':
        rt = 0.035
    elif row['sh_funnelstartdepth']=='7) Quote':
        rt = 0.054
    elif row['sh_funnelstartdepth']=='7) Sweepstakes':
        rt = 0.01
    else:
        rt = 0.029
    return rt

def preprocess_for_engine(alldata):

    # DEBUG
    #print(alldata[:,:])

    # subset for only Canada and United States
    alldata = alldata[(alldata.country == 'Canada') | (alldata.country ==
'United States')]

    # create starting_page_depth column
    alldata['starting_page_depth'] =
np.where(pd.isnull(alldata['s_startingpage']), 0,

alldata['s_startingpage'].str.count('/'))

    # create canada flag
    alldata['canada_flag'] = np.where(alldata['country'] == 'Canada', 1,
0)

    # create device flags and conversion rate column
    alldata['desktop_flag'] = np.where(alldata['devicetype'] == 'desktop',
1, 0)
    alldata['mobile_flag'] = np.where(alldata['devicetype'] == 'mobile',
1, 0)
    alldata['tablet_flag'] = np.where(alldata['devicetype'] == 'tablet',
1, 0)

    alldata['device_conv_rt'] = alldata.apply(dev, axis=1)

    # create brand flags and conversion rate column
    alldata['ACE_flag'] = np.where(alldata['brand'] == 'ACE', 1, 0)
    alldata['ATV_flag'] = np.where(alldata['brand'] == 'ATV', 1, 0)
    alldata['GEM_flag'] = np.where(alldata['brand'] == 'GEM', 1, 0)
    alldata['GRL_flag'] = np.where(alldata['brand'] == 'GRL', 1, 0)
```

```
    alldata['IND_flag'] = np.where(alldata['brand'] == 'IND', 1, 0)
    alldata['ORV_flag'] = np.where(alldata['brand'] == 'ORV', 1, 0)
    alldata['RGR_flag'] = np.where(alldata['brand'] == 'RGR', 1, 0)
    alldata['RZR_flag'] = np.where(alldata['brand'] == 'RZR', 1, 0)
    alldata['SLG_flag'] = np.where(alldata['brand'] == 'SLG', 1, 0)
    alldata['SNO_flag'] = np.where(alldata['brand'] == 'SNO', 1, 0)

    alldata['brand_conv_rt'] = alldata.apply(brn, axis=1)

    # create channel flags and conversion rate column
    alldata['chan_direct'] = np.where(alldata['channel'] == 'Direct', 1,
0)
    alldata['chan_display_banner'] = np.where(alldata['channel'] ==
'Display / Banner', 1, 0)
    alldata['chan_email'] = np.where(alldata['channel'] == 'Email', 1, 0)
    alldata['chan_fb'] = np.where(alldata['channel'] == 'FB Ads to Web',
1, 0)
    alldata['chan_organic'] = np.where(alldata['channel'] == 'Organic', 1,
0)
    alldata['chan_paid'] = np.where(alldata['channel'] == 'Paid Search',
1, 0)
    alldata['chan_ref'] = np.where(alldata['channel'] == 'Referral', 1, 0)
    alldata['chan_remark'] = np.where(alldata['channel'] == 'Remarketing',
1, 0)
    alldata['chan_retarg'] = np.where(alldata['channel'] == 'Retargeting',
1, 0)

    alldata['channel_conv_rt'] = alldata.apply(chn, axis=1)

    # create funnel flags and conversion rate column
    alldata['fun_phase1'] = np.where(alldata['sh_funnelstartdepth'] == '1)
Phase 1: Brand Exploration (140 Days)', 1, 0)
    alldata['fun_phase3'] = np.where(alldata['sh_funnelstartdepth'] == '3)
Phase 3: Model Details (100 Days)', 1, 0)
    alldata['fun_phase4'] = np.where(alldata['sh_funnelstartdepth'] == '4)
Phase 4: Customize and Price (75 Days)', 1,
                                     0)
    alldata['fun_quoteform6'] = np.where(alldata['sh_funnelstartdepth'] ==
'6) Quote Form', 1, 0)
    alldata['fun_rideform6'] = np.where(alldata['sh_funnelstartdepth'] ==
'6) Test Ride Form', 1, 0)
    alldata['fun_custquote7'] = np.where(alldata['sh_funnelstartdepth'] ==
'7) Customize Quote', 1, 0)
    alldata['fun_quote7'] = np.where(alldata['sh_funnelstartdepth'] == '7)
Quote', 1, 0)
    alldata['fun_sweep7'] = np.where(alldata['sh_funnelstartdepth'] == '7)
Sweepstakes', 1, 0)

    alldata['funnel_conv_rt'] = alldata.apply(fnl, axis=1)

    # create column of seconds from start of day, sin and cos
    alldata['secs_start_day'] =
pd.to_numeric(alldata.groupby('date')['visitstarttime'].transform(lambda
x: x - x.min()))
```

```
    alldata['secs_day_cos'] = np.cos((2 * math.pi / 85400) *
alldata['secs_start_day'])
    alldata['secs_day_sin'] = np.sin((2 * math.pi / 85400) *
alldata['secs_start_day'])

    # take date and get (1) days since start of year, sin and cos (2) day
of week (3) day of month
    alldata.date = alldata.date.astype(str)  # convert date field to
string
    # (1)
    alldata['days_start_year'] = pd.to_numeric(alldata.apply(
        lambda x: ((pd.to_datetime(x['date'], format='%Y%m%d') -
pd.to_datetime(x['date'][:4] + '0101')).days + 1),
        axis=1))
    alldata['days_year_cos'] = np.cos((2 * math.pi / 365) *
alldata['days_start_year'])
    alldata['days_year_sin'] = np.sin((2 * math.pi / 365) *
alldata['days_start_year'])
    # (2)
    alldata['day_wk'] = pd.to_datetime(alldata['date'],
format='%Y%m%d').dt.dayofweek
    # (3)
    alldata['day_mnth'] = pd.to_datetime(alldata['date'],
format='%Y%m%d').dt.day

    # create dealer_conv_rate column
    alldata['dealer_conv_rt'] = np.where(alldata['maxdealerid'] == np.NaN,
0.009, 0.048)

    # bring in dealer data and geodemographics
    dealers = pd.read_csv('lead_score_engine/dealer data inputs
20171024.csv', encoding="cp1252")
    geodems = pd.read_csv('lead_score_engine/geodemographics.csv',
encoding="cp1252")
    canpostals = pd.read_csv('lead_score_engine/canada postal codes.csv',
encoding="cp1252")

    # remove all letter data from the max dealer id
    alldata.maxdealerid = alldata.maxdealerid.astype(str)
    alldata['maxdealerid'] = alldata.apply(lambda x: re.sub("[^0-9]", "0",
x['maxdealerid']), axis=1)

    # make sure each of the join columns are types to be joinable
    alldata.maxdealerid = alldata.maxdealerid.astype(float)
    dealers.dealerid = dealers.dealerid.astype(float)
    canpostals.postal_code = canpostals.postal_code.astype(str)
    alldata.postalcode = alldata.postalcode.astype(str)
    geodems.ZIP = geodems.ZIP.astype(str)
    canpostals.postal_code = canpostals.postal_code.astype(str)

    # merge lead data with dealer data
    maintable = pd.merge(alldata, dealers, how='left',
left_on=['maxdealerid'], right_on=['dealerid'])
```

```
    # now merge data with US geodemographics
    maintable = pd.merge(maintable, geodems, how='left',
left_on=['postalcode'], right_on=['ZIP'])

    # create merge columns for Canada data frame
    maintable['postal_first3'] = [item[:3] for item in
maintable.postalcode]
    canpostals['postal_first3'] = [item[:3].lower() for item in
canpostals.postal_code]

    # merge maintable and canada postalcodes together for final table
    maintable = pd.merge(maintable, canpostals, how='left',
on=['postal_first3'])

    # replace NAs in lead/dealer longitude and latitude columns with
values from read in data
    maintable['latitude'] = maintable['latitude'].fillna(maintable['lat'])
# lead latitude from US geodemos
    maintable['longitude'] =
maintable['longitude'].fillna(maintable['lng'])  # lead longitude from US
geodemos
    maintable['dealerLatitude'] = maintable['dealerLatitude'].fillna(
        maintable['lat'])  # dealer latitude from US geodemos
    maintable['dealerLongitude'] = maintable['dealerLongitude'].fillna(
        maintable['lng'])  # dealer longitude from US geodemos

    maintable['latitude'] =
maintable['latitude'].fillna(maintable['Latitude'])  # lead latitude from
CA postal codes
    maintable['longitude'] = maintable['longitude'].fillna(
        maintable['Longitude'])  # lead longitude from CA postal codes
    maintable['dealerLatitude'] = maintable['dealerLatitude'].fillna(
        maintable['Latitude'])  # dealer latitude from CA postal codes
    maintable['dealerLongitude'] = maintable['dealerLongitude'].fillna(
        maintable['Longitude'])  # dealer longitude from CA postal codes

    # replace 0s in lead/dealer lat long with NaN
    maintable['latitude'] = maintable['latitude'].replace(0, np.NaN)
    maintable['longitude'] = maintable['longitude'].replace(0, np.NaN)
    maintable['dealerLatitude'] = maintable['dealerLatitude'].replace(0,
np.NaN)
    maintable['dealerLongitude'] = maintable['dealerLongitude'].replace(0,
np.NaN)

    # change positive longitudes to negative
    maintable['longitude'] = -abs(pd.to_numeric(maintable['longitude']))
    maintable['dealerLongitude'] = -
abs(pd.to_numeric(maintable['dealerLongitude']))

    # create distance to dealer column
    maintable['dist_to_dealer'] = maintable.apply(
        lambda x: great_circle((x['latitude'], x['longitude']),
(x['dealerLatitude'], x['dealerLongitude'])).miles,
        axis=1)
```

```
    # create dealer activity column
    maintable['dealer_activity'] = maintable.apply(
        lambda x: (x['ACE_flag'] * x['ORVsalesmult']) + (x['ATV_flag'] *
x['ORVsalesmult']) + (
        x['GEM_flag'] * x['GEMsalesmult']) + (x['GRL_flag'] *
x['ORVsalesmult']) + (
                  x['IND_flag'] * x['INDsalesmult']) + (x['ORV_flag'] *
x['ORVsalesmult']) + (
                  x['RGR_flag'] * x['ORVsalesmult']) + (x['RZR_flag'] *
x['ORVsalesmult']) + (
                  x['SLG_flag'] * x['SLGsalesmult']) + (x['SNO_flag'] *
x['SNOsalesmult']), axis=1)

    # remove identifiers
    visit_id = maintable.pop('visitstarttime')
    cs_id = maintable.pop('csid')

    # drop all unneeded columns
    maintable.drop(['sh_funnelstartdepth'
                       , 'channel'
                       , 'brand'
                       , 'devicetype'
                       , 'country'
                       , 's_startingpage'
                       , 'end_dt'
                       , 'region'
                       , 'city'
                       , 's_startingpage'
                       , 'start_dt'
                       , 'date'], axis=1, inplace=True)
    maintable.drop(['dealerid'
                       , 'dealerpostalcode'
                       , 'dealerLatitude'
                       , 'dealerLongitude'
                       , 'ORVsalesmult'
                       , 'SLGsalesmult'
                       , 'SNOsalesmult'
                       , 'GEMsalesmult'
                       , 'INDsalesmult'
                       , 'ZIP'
                       , 'postalcode'
                       , 'maxdealerid'
                       , 'mindealerid'
                       , 'lat'
                       , 'lng'
                       , 'postal_first3'
                       , 'postal_code'
                       , 'Latitude'
                       , 'Longitude'], axis=1, inplace=True)

    # replace NAs with zeros
    impute_zeros = ['s_timeonsite', 's_totalbounces', 's_totalpageviews',
'newvisits']
```

```
    for col in impute_zeros:
        maintable[col] = maintable[col].replace(np.NaN, 0)

    return maintable,visit_id, cs_id

def run_lead_engine(maintable,visit_id, cs_id):
    gbm = pickle.load(open("lead_score_engine/leadscoring.pickle.dat",
"rb"))
    # predict on the leads
    gbm_preds = gbm.predict_proba(maintable)

    # create ouput file and read to the directory
    #gbm_preds_out = np.column_stack((visit_id, cs_id, gbm_preds[:, 1]))
    #np.savetxt('lead_score_engine/lead_probabilities.csv', gbm_preds_out,
delimiter=',', fmt='%s')  # change file name
    #print ("Lead Probabilities saved in
lead_score_engine/lead_probabilities.csv file.")

    maintable["visitstarttime"] = visit_id
    maintable["csid"] = cs_id
    maintable["Preds"] = gbm_preds[:, 1]

    return maintable


def adjust_for_predictor(maintable):
    # RJS commmented this out: not necessary to remove columns right
before subsetting columns, it is redundant (and it causing errors)
    #maintable.drop(['PurchaseBrand', 'PurchaseCSID',
'PurchaseConsumerID', 'PurchaseDate', 'PurchaseMarketingName',
    #                'PurchaseModelDescription', 'PurchaseName',
'PurchaseVIN', 'Purchasecreatedon'], axis=1)
    l1 = ['pageviews', 'sh_sweepstakescount', 'sh_emailsignup01',
'sh_getquotecount', 'sh_testridecount',
          'sh_customquotecount', 'sh_tradeincount', 's_ecommpages',
's_homepages', 's_specialofferspages',
          's_dealerlocator', 's_brandpages', 's_customize',
's_modelpages', 's_timeonsite', 's_totalbounces',
          's_totalpageviews', 'latitude', 'longitude', 'newvisits',
'uniquedealercount', 'ps_sessions', 'ps_pageviews',
          'psh_sweepstakescount', 'psh_emailsignupcount',
'psh_getquotecount', 'psh_testridecount',
          'psh_customquotecount', 'psh_tradeincount', 'ps_ecommpages',
'ps_homepages', 'ps_specialofferspages',
          'ps_dealerlocatorpages', 'ps_brandpages', 'ps_customizepages',
'ps_modelpages', 'starting_page_depth',
          'canada_flag', 'desktop_flag', 'mobile_flag', 'tablet_flag',
'device_conv_rt', 'ACE_flag', 'ATV_flag',
          'GEM_flag', 'GRL_flag', 'IND_flag', 'ORV_flag', 'RGR_flag',
'RZR_flag', 'SLG_flag', 'SNO_flag',
          'brand_conv_rt', 'chan_direct', 'chan_display_banner',
'chan_email', 'chan_fb', 'chan_organic', 'chan_paid',
          'chan_ref', 'chan_remark', 'chan_retarg', 'channel_conv_rt',
'fun_phase1', 'fun_phase3', 'fun_phase4',
```

```
          'fun_quoteform6', 'fun_rideform6', 'fun_custquote7',
'fun_quote7', 'fun_sweep7', 'funnel_conv_rt',
          'secs_start_day', 'secs_day_cos', 'secs_day_sin',
'days_start_year', 'days_year_cos', 'days_year_sin',
          'day_wk', 'day_mnth', 'dealer_conv_rt', 'suspended_flag',
'Input_#1_ATV', 'Input_#1_RGR', 'Input_#1_RZR',
          'COM', 'S0', 'VIC', 'IND', 'LEV', 'SLG', 'BOA', 'CRP', 'GEN',
'GPL', 'MIL', 'OSP', 'PGA', 'PWC', 'TSL',
          'ORVslstotal', 'SLGslstotal', 'SNOslstotal', 'GEMslstotal',
'INDslstotal', 'Right_ATV', 'SXS',
          'Right_Right_SNO', 'MCY', 'SCT', 'Right_Right_PWC', 'MARINE',
'Right_Right_PGA', 'Honda', 'Triumph', 'Suzuki',
          'Ducati', 'ArcticCat', 'BMW', 'Polaris', 'CFMoto', 'Kawasaki',
'Harley', 'BRP', 'Yamaha', 'POPULATION',
          'HH_INCOME', 'PC_INCOME', 'HOUSE_VAL', 'URBAN', 'SUBURBAN',
'FARM', 'NONFARM', 'WHITE', 'BLACK', 'INDIAN',
          'ASIAN', 'HAWAIIAN', 'RACE_OTHER', 'HISPANIC', 'AGE_0_4',
'AGE_5_9', 'AGE_10_14', 'AGE_15_17', 'AGE_18_19',
          'AGE_20', 'AGE_21', 'AGE_22_24', 'AGE_25_29', 'AGE_30_34',
'AGE_35_39', 'AGE_40_44', 'AGE_45_49', 'AGE_50_54',
          'AGE_55_59', 'AGE_60_61', 'AGE_62_64', 'AGE_65_66', 'AGE_67_69',
'AGE_70_74', 'AGE_75_79', 'AGE_80_84',
          'AGE_85_PLS', 'EDU_LESS9', 'EDU_9_12', 'EDU_HIGHSC',
'EDU_SOMECL', 'EDU_ASSOC', 'EDU_BACH', 'EDU_PROF',
          'dist_to_dealer', 'dealer_activity']

    df = pd.DataFrame()
    df = pd.concat([df, maintable[l1]], axis=1)
    return df


def start(leadData):
    #print ("Merging all sources ...")
    #alldata=merge_all_sources()
    #print ("Preprocessing data for lead score engine")

    # DEBUG: update pandas options so we can see all the columns in the
table
    pd.set_option("display.max_columns",999)

    # creating initial table
    maintable, visit_id, cs_id=preprocess_for_engine(leadData)

    data={"maintable":maintable,
          "visit_id":visit_id,
          "cs_id":cs_id}

    with open("asdfg.pkl","wb") as fp:
        pickle.dump(data,fp)

    maintable=adjust_for_predictor(maintable)


    # DEBUG
```

```
    #print( maintable.sample(5))
    #( maintable.dtypes)

    # all columns are numeric values, but some are stored as text, so
convert all to numeric
    maintable = maintable.apply(pd.to_numeric)

    # DEBUG
    #print( maintable.sample(5))
    #print( maintable.dtypes)


    #print("Done Pre processing")
    leadScores = run_lead_engine(maintable,visit_id, cs_id)
    return leadScores
```