Kaggle Code Competition - Time Series Project

M5 Forecasting - Accuracy Estimate the unit sales of Walmart retail goods

Claudia Wang

Introduction

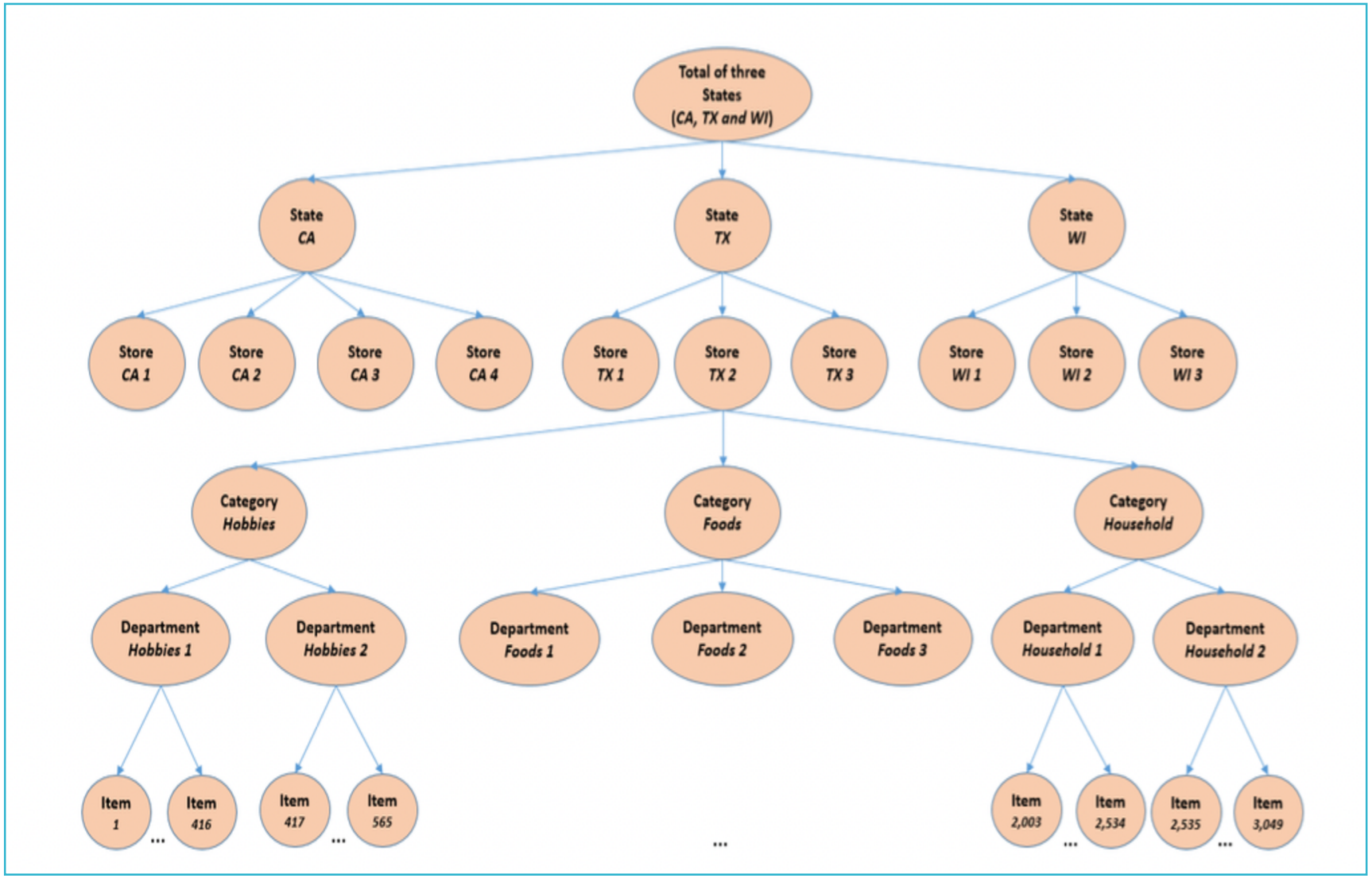
- This is a code competition hold by Kaggle (a community of data scientists) and the University of Nicosia.
- The objective of this competition is to use hierarchical sales data from Walmart to forecast daily sales for the next 28 days. It challenged us to improve forecast accuracy.
- Metric Used for evaluation: Weighted Root Mean Squared Scaled Error (RMSSE).
- My team win the bronze metal, which is top 7% of the 5,558 teams.

Data Overview

- The data, covers stores in three US States (California, Texas, and Wisconsin) and includes item level, department, product categories, and store details. In addition, it has explanatory variables such as price, promotions, day of the week, and special events.

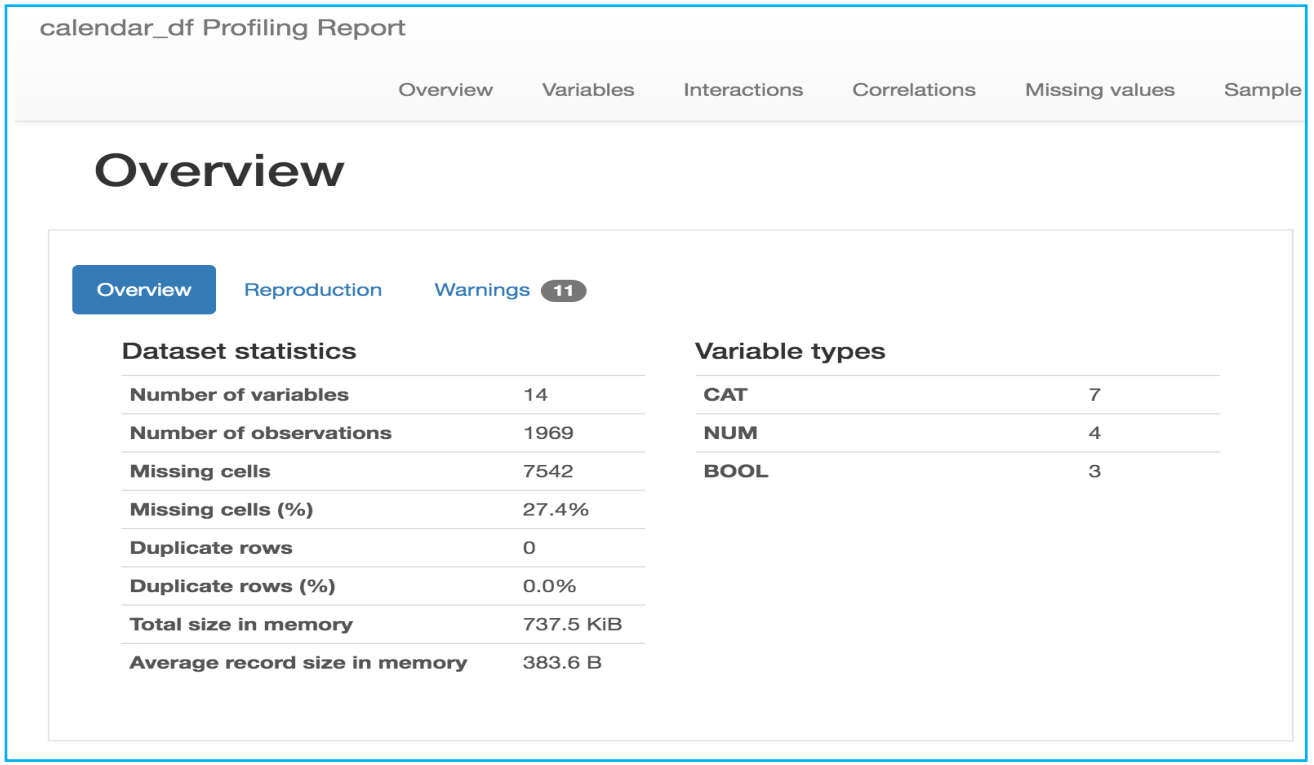
- Datasets:
  - **calendar.csv** - Contains information about the dates on which the products are sold.
  - **sales_train_validation.csv** - Contains the historical daily unit sales data per product and store [d_1 - d_1913]
  - **sample_submission.csv** - The correct format for submissions.
  - **sell_prices.csv** - Contains information about the price of the products sold per store and date.
  - **sales_train_evaluation.csv** - Includes sales [d_1 - d_1941] (labels used for the Public leaderboard)

Data Overview



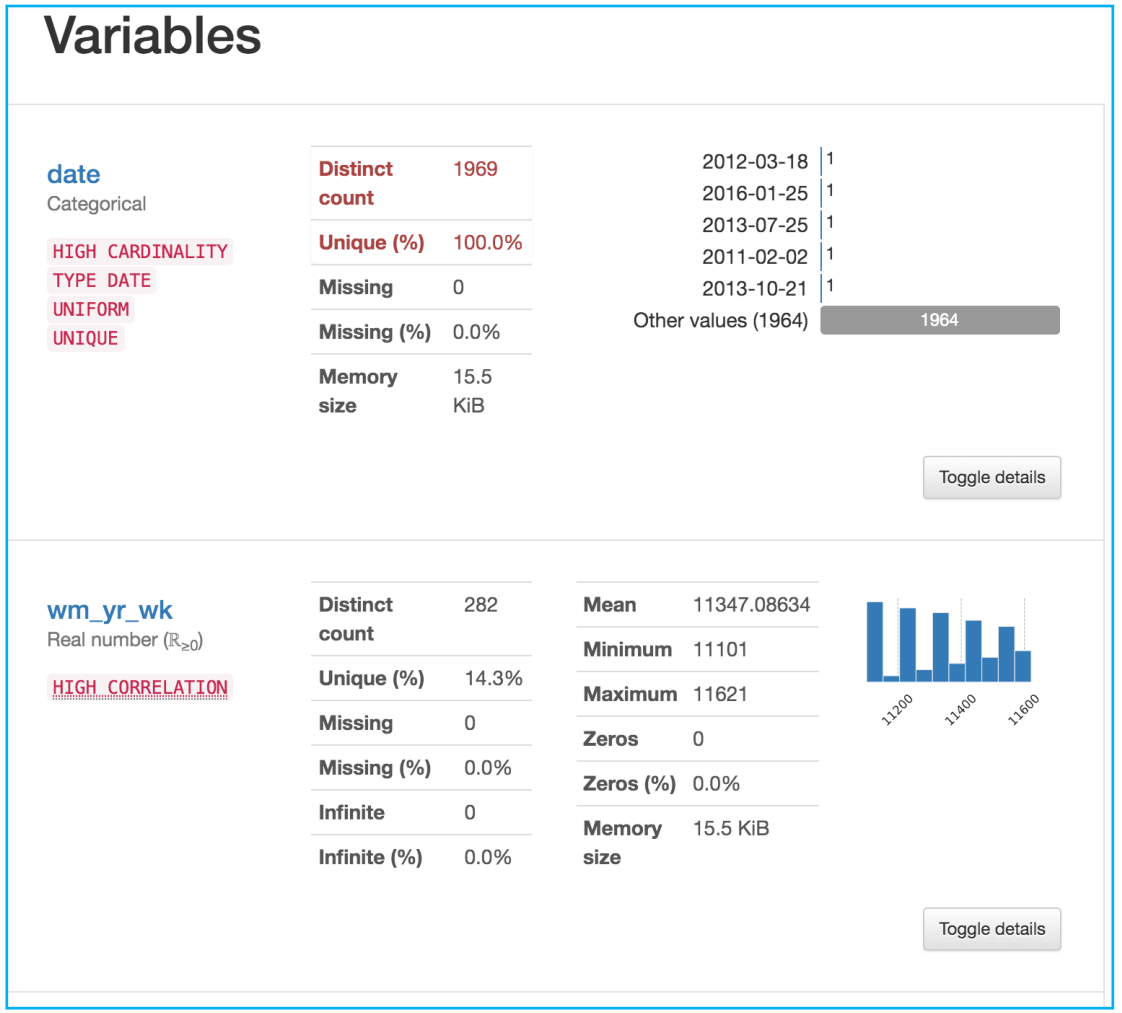
Exploratory Data Analysis

- Create DataFrame and build profiling report.
  - DataFrame : sell_prices_df, calendar_df, sales_train_validation_df, submission_df
- Profiling Report sample:



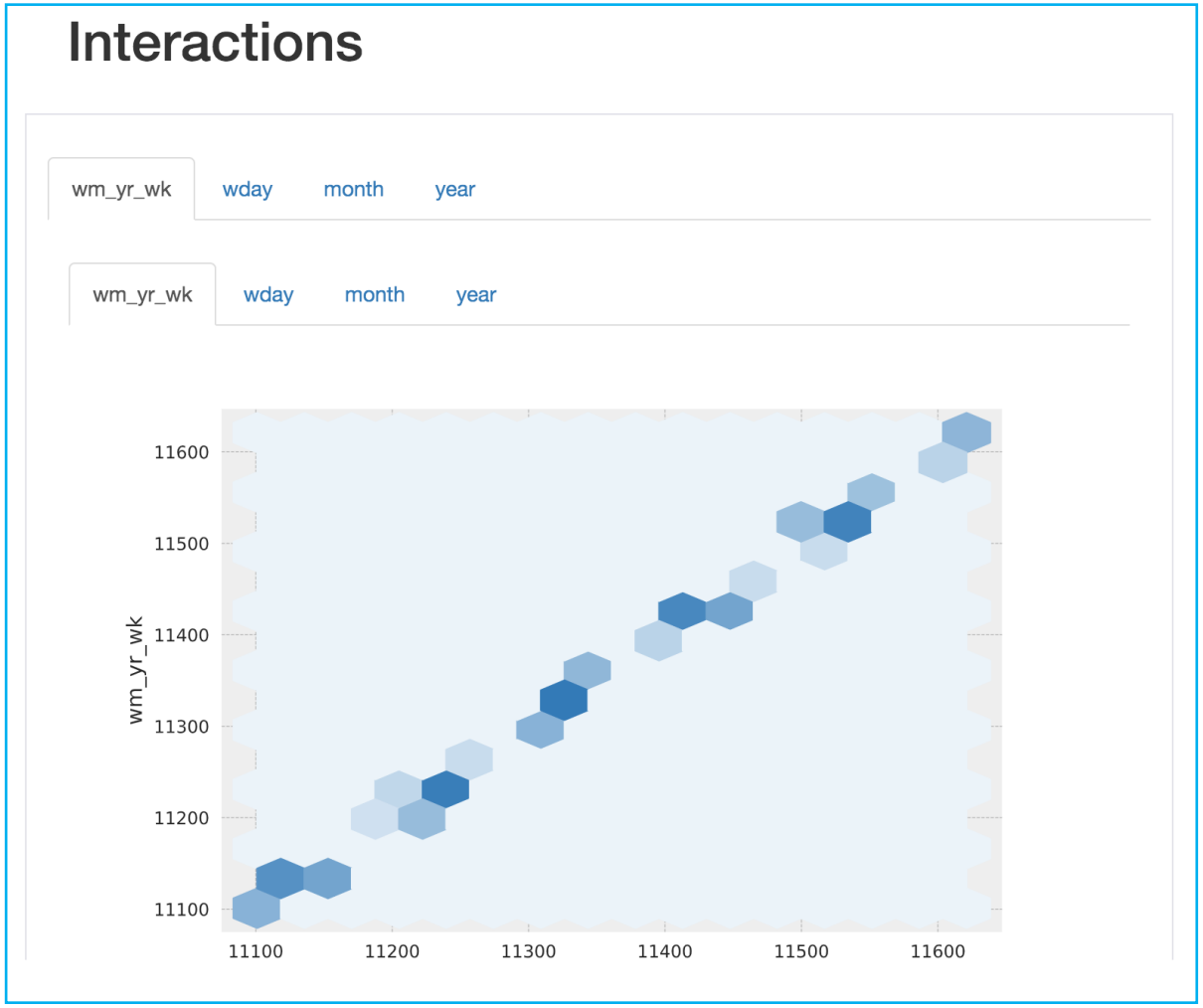
Exploratory Data Analysis

- Calender_df Profiling sample:



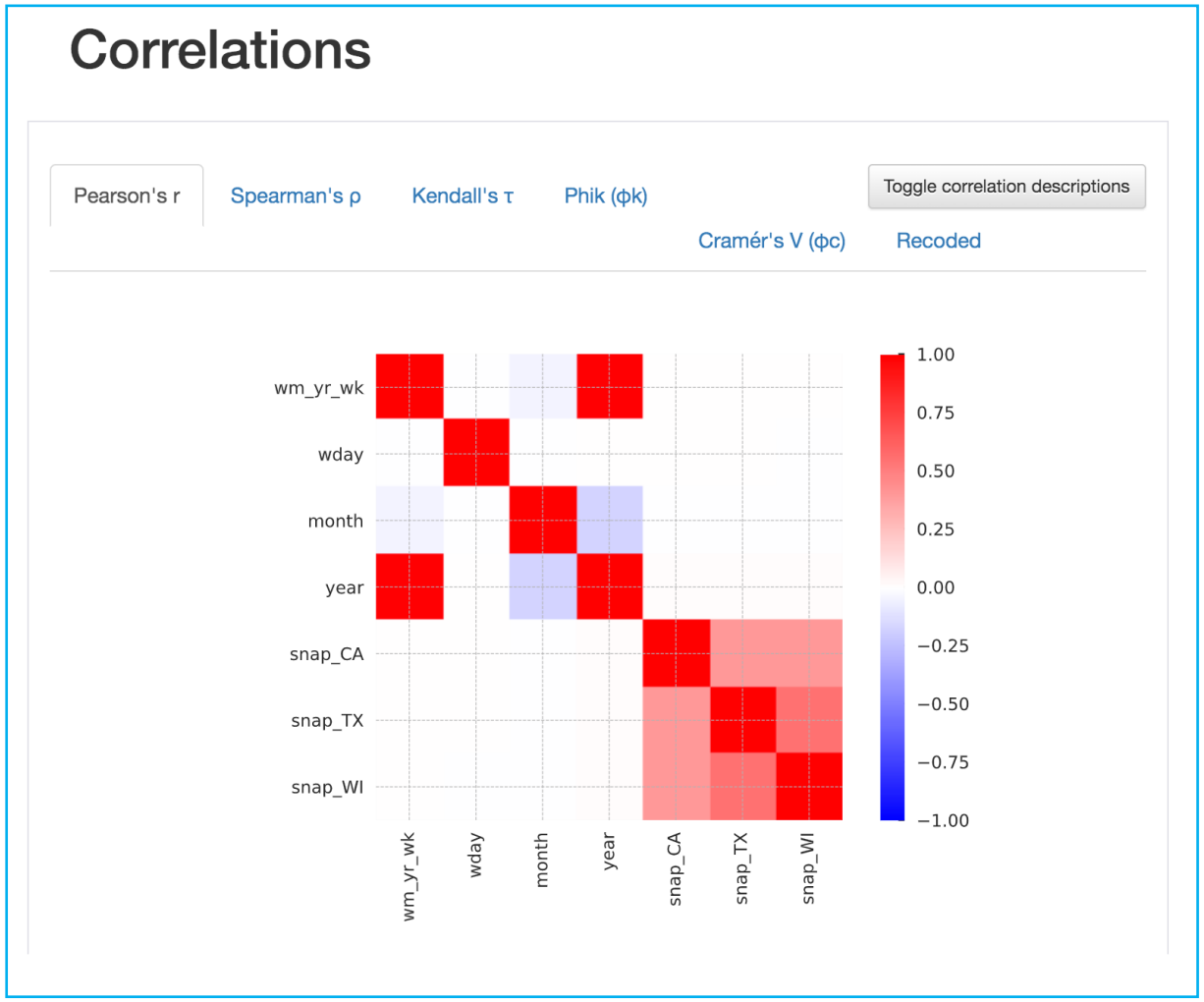
Exploratory Data Analysis

- Calender_df Profiling sample:



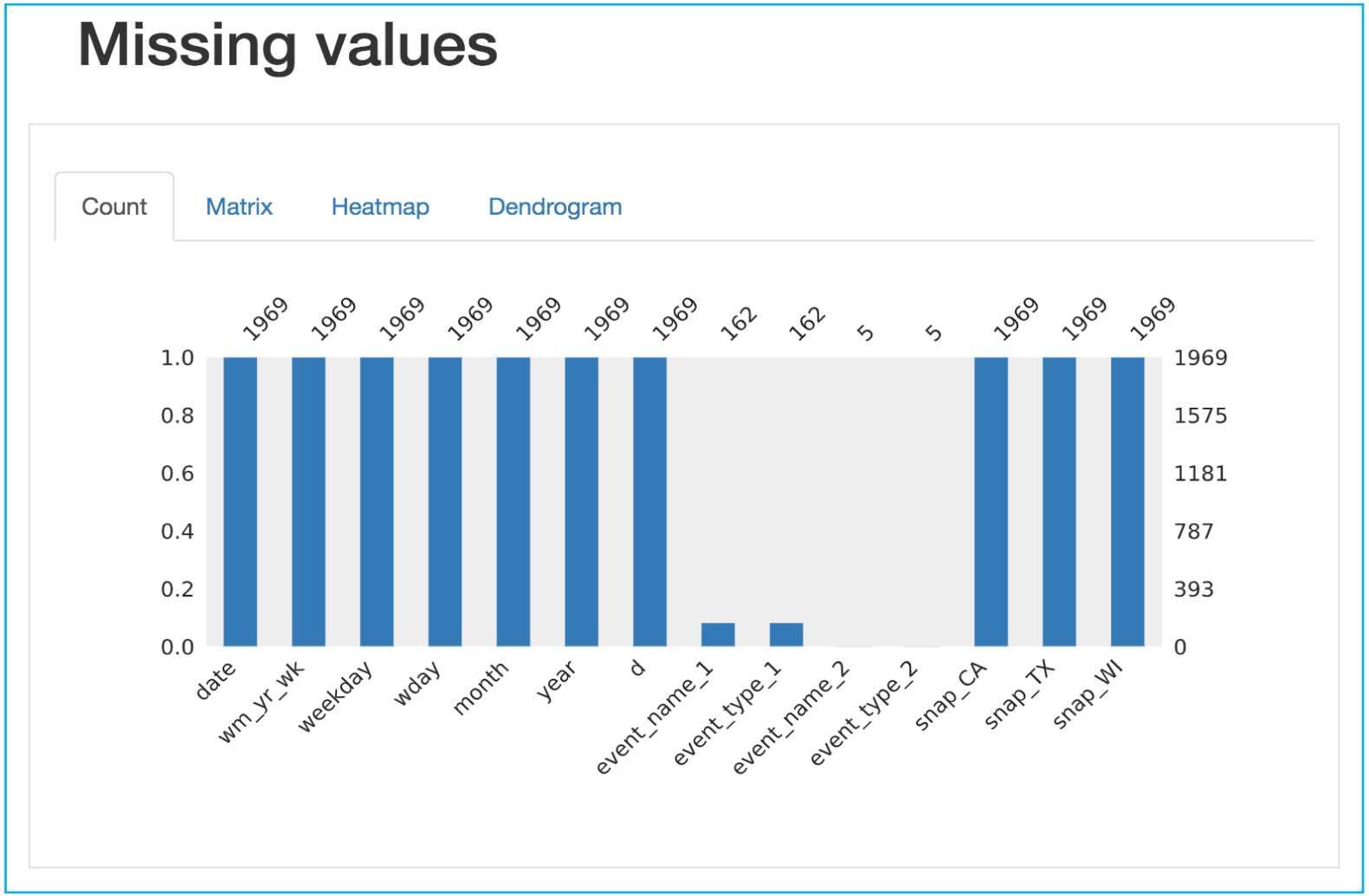
Exploratory Data Analysis

- Calender_df Profiling sample:



Exploratory Data Analysis

- Calender_df Profiling sample:



Exploratory Data Analysis

- Calender_df Profiling sample:

Sample

First rows

	date	wm_yr_wk	weekday	wday	month	year	d	event_name_1	event_type_1
0	2011-01-29	11101	Saturday	1	1	2011	d_1	NaN	NaN
1	2011-01-30	11101	Sunday	2	1	2011	d_2	NaN	NaN
2	2011-01-31	11101	Monday	3	1	2011	d_3	NaN	NaN
3	2011-02-01	11101	Tuesday	4	2	2011	d_4	NaN	NaN
4	2011-02-02	11101	Wednesday	5	2	2011	d_5	NaN	NaN
5	2011-02-03	11101	Thursday	6	2	2011	d_6	NaN	NaN
6	2011-02-04	11101	Friday	7	2	2011	d_7	NaN	NaN
7	2011-02-05	11102	Saturday	1	2	2011	d_8	NaN	NaN
8	2011-02-06	11102	Sunday	2	2	2011	d_9	SuperBowl	Sporting
9	2011-02-07	11102	Monday	3	2	2011	d_10	NaN	NaN

Exploratory Data Analysis

- **Calender_df findings:**
  - date: there are 1969 different dates, which shows this df has data for 1969 different dates.
  - d: there are 1969 different d values (Also d has lots of unique values that's why it has `HIGH CARDINALITY`)
  - event_name and event_type columns denotes special promotional events thus event_name_1, event_type_1, event_name_2, event_type_2 contains a lot of missing values.
  - date features has very corelation and features snap_CA, snap_TX and snap_WI also show some corelation between them.

Exploratory Data Analysis

- Plot sales of random 10 items on actual dates:

Exploratory Data Analysis

- Plot sales of random 10 items on actual dates:
- **Findings:**
  - Observations: It is common to see an item unavailable for a period of time.
  - Some items only sell 1 or less in a day, making it very hard to predict.
  - Other items show spikes in their demand (super bowl Sunday).

Feature Engineering

- **Dataset preprocess:**
  - Downcast in order to save memory;
  - Label Encode categorical variable;
  - Leave NaN as it is;
  - Melt sales data;
  - Merge data frames;
  - Drop redundant calendar features;
  - Calculate weight for RMSSE.

Feature Engineering

- **Add new features and rolling:**
  - Demand Feature:
    - Shift 1, 2, 3 days Features;
    - Shift 'DAYS_PRED', rolling size [7, 15, 50, 30, 21, 60, 90, 180] then calculate Standard Deviation and Mean Features;
    - Shift 'DAYS_PRED', rolling size 30 then calculate Skew and Kurt Features;
  - Sell Feature:
    - Shift 1, 2, 3 days Features;
    - Shift 'DAYS_PRED', rolling size [7, 15, 50, 30, 21, 60, 90, 180] then calculate Standard Deviation and Mean Features;
    - Shift 'DAYS_PRED', rolling size 30 then calculate Skew and Kurt Features;
- **Fill NaN features with unknown**

Feature Engineering

```
[ ] # All the features after feature engineering
    features = ["item_id", "dept_id", "cat_id", "store_id", "state_id",
      "event_name_1", "event_type_1", "event_name_2", "event_type_2", "snap_CA", "snap_TX", "snap_WI", "sell_price",
      "id_shift_t28","id_shift_t29", "id_shift_t30",
      'id_store_id_shift_t28', 'id_store_id_shift_t29', 'id_store_id_shift_t30',
      'id_store_id_rolling_std_t7', 'id_store_id_rolling_std_t30','id_store_id_rolling_std_t180',
      'id_store_id_rolling_mean_t7', 'id_store_id_rolling_mean_t30', 'id_store_id_rolling_mean_t180',
      'id_store_id_rolling_skew_t30', 'id_store_id_rolling_kurt_t30',
      "id_rolling_std_t7", "id_rolling_std_t30", "id_rolling_std_t60", "id_rolling_std_t90", "id_rolling_std_t180",
      "id_rolling_mean_t7", "id_rolling_mean_t30", "id_rolling_mean_t60", "id_rolling_mean_t90", "id_rolling_mean_t180",
      "id_rolling_skew_t30", "id_rolling_kurt_t30",
      "price_change_t1", "price_change_t365",
      "rolling_price_std_t7", "rolling_price_std_t30",
      "year", "month","week","day","dayofweek","is_year_end","is_year_start","is_month_end","is_month_start","is_weekend",
      'id_rolling_std_t21', 'id_rolling_mean_t21',
      'id_store_id_rolling_std_t60', 'id_store_id_rolling_std_t90',
      'id_store_id_rolling_mean_t60', 'id_store_id_rolling_mean_t90']
```

```
[ ] categorical_features = ['item_id', 'dept_id', 'cat_id',
     'store_id', 'state_id',
     'event_name_1', 'event_type_1', 'event_name_2','event_type_2',
     'snap_CA','snap_TX', 'snap_WI',]
```

Modelling and Prediction

- **Create LightGBM Model:**

```
def run_lgb(data, wrmsse):
    feature_importance = pd.DataFrame()

    mask1 = data['date']<= '2016-04-24'
    mask2 = data['date']> data['date'].min() + timedelta(208)
    mask = mask1&mask2
    x_train = data[mask]
    y_train = x_train['demand']
    x_val = data[(data['date'] > '2016-04-24') & (data['date'] <= '2016-05-22')]
    y_val = x_val['demand']
    test = data[(data['date'] >=  (datetime(2016,5,23) - timedelta(100)))]
    test = test.reset_index(drop = True)
    keep_cols = [col for col in test if '_tmp_' not in col]
    test[keep_cols].to_pickle('input/test/test_data_private.pkl')

    gc.collect()
    params = {
                'boosting_type': 'gbdt',
                'objective': 'tweedie',
                'tweedie_variance_power': 1.1,
                'metric': 'rmse',
                'subsample': 0.5,
                'subsample_freq': 1,
                'learning_rate': 0.03,
                'num_leaves': 2**11-1,
                'min_data_in_leaf': 2**12-1,
                'feature_fraction': 0.5,
                'max_bin': 100,
#                     'n_estimators': 1400,
                'boost_from_average': False,
                'seed':1992,
                'n_jobs':-1,
                'verbose': 1,
            }

    train_set = lgb.Dataset(x_train[features], label = y_train, weight = x_train['weight'])
    val_set = lgb.Dataset(x_val[features], y_val)

    del x_train, y_train
```

Modelling and Prediction

- **Calculate WRMSSE**

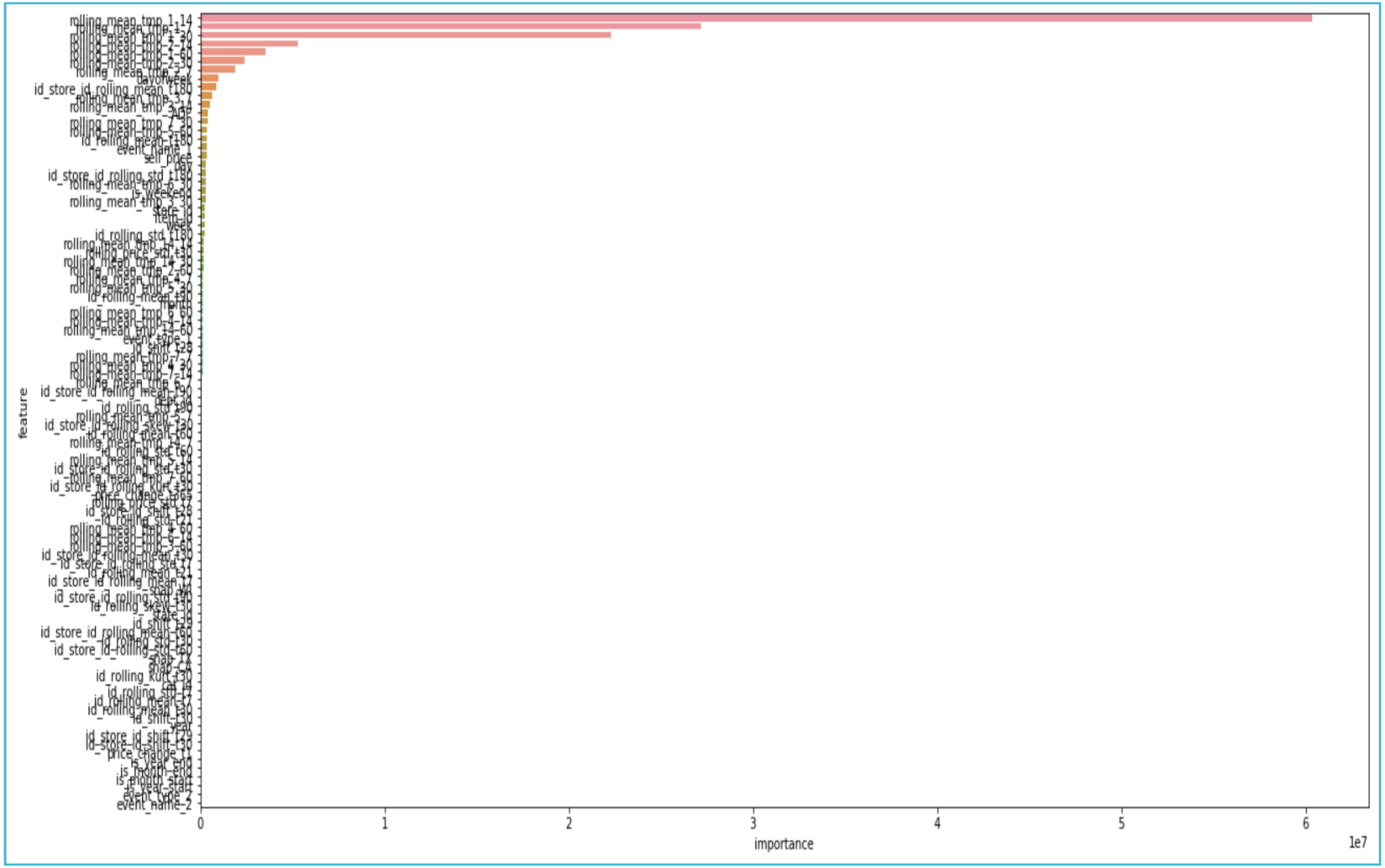
$$WRMSSE = \sum_{i=1}^{42,800}\left(W_i \times \sqrt{\frac{\sum_{j=1}^{28}(Y_t - \hat{Y}_t)^2}{S_i}}\right)$$

$$S_i = \frac{1}{n-1}\sum_{t=2}^{n}(Y_t - Y_{t-1})^2$$

```
Training until validation scores don't improve for 500 rounds.
[100]   training's rmse: 2.09956        training's wrmsse: 0.616345     valid_1's rmse: 2.00886 valid_1's wrmsse: 0.600455
[200]   training's rmse: 2.04695        training's wrmsse: 0.48492      valid_1's rmse: 1.9749  valid_1's wrmsse: 0.5069
[300]   training's rmse: 2.02603        training's wrmsse: 0.441164     valid_1's rmse: 1.96675 valid_1's wrmsse: 0.495198
[400]   training's rmse: 2.01382        training's wrmsse: 0.425763     valid_1's rmse: 1.96324 valid_1's wrmsse: 0.493084
[500]   training's rmse: 2.00369        training's wrmsse: 0.415422     valid_1's rmse: 1.96025 valid_1's wrmsse: 0.491965
[600]   training's rmse: 1.99546        training's wrmsse: 0.409564     valid_1's rmse: 1.9585  valid_1's wrmsse: 0.492117
[700]   training's rmse: 1.98806        training's wrmsse: 0.404592     valid_1's rmse: 1.95709 valid_1's wrmsse: 0.493056
[800]   training's rmse: 1.98121        training's wrmsse: 0.400914     valid_1's rmse: 1.95571 valid_1's wrmsse: 0.493229
[900]   training's rmse: 1.97446        training's wrmsse: 0.3975       valid_1's rmse: 1.95439 valid_1's wrmsse: 0.494564
Early stopping, best iteration is:
[483]   training's rmse: 2.00547        training's wrmsse: 0.417348     valid_1's rmse: 1.96108 valid_1's wrmsse: 0.491448
Our val rmse score is 1.9610817118899775
Our val wrmsse score is 0.491447550353983
Wall time: 2h 12min 43s
```

Modelling and Prediction

- **Plot feature importance**

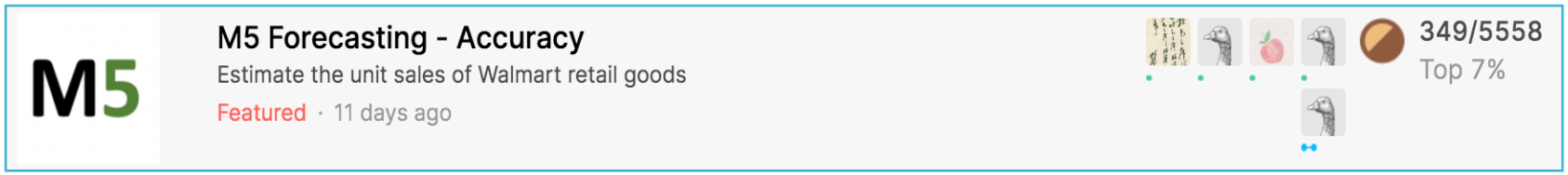


Modelling and Prediction

- **Create csv submission;**

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
1	id	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20	F21	F22	F23	F24	F25	F26	F27	F28
2	HOBBIES_1_001_CA_1_validation	0.7624373	0.7061886	0.7193367	0.7422777	0.9269432	1.0176128	1.1434069	0.8475899	0.8566439	0.7635918	0.7435389	0.8932477	1.0665661	0.8385692	0.8495709	0.8181542	0.7498209	0.7345324	0.7710561	0.9578873	0.9789015	0.8257689	0.7694697	0.7348744	0.7851125	0.8908773	1.0503519	0.9592945
3	HOBBIES_1_002_CA_1_validation	0.194053906	0.1866352	0.1819556	0.1780171	0.222009	0.2649614	0.2942155	0.2365784	0.2230528	0.1864661	0.2291339	0.2226904	0.226308	0.2212184	0.2255886	0.2003327	0.1827391	0.1874519	0.2078382	0.2267397	0.228947	0.1982347	0.1809399	0.1883979	0.1816606	0.1959119	0.236309	0.2523735
4	HOBBIES_1_003_CA_1_validation	0.450539736	0.4193501	0.4166848	0.4094886	0.5783966	0.7430134	0.7578153	0.4915435	0.4962625	0.4427821	0.4718659	0.5309782	0.6760972	0.5462531	0.4432753	0.4184354	0.4254525	0.4612618	0.5498805	0.7113861	0.7155298	0.452633	0.4240501	0.4523251	0.4834415	0.6180377	0.7356987	0.6958159
5	HOBBIES_1_004_CA_1_validation	1.637289278	1.2698669	1.3254814	1.4930732	1.9083821	2.8794627	3.1708963	1.7620781	1.4811849	1.4103024	1.5700365	1.76021	2.9129444	2.2139131	1.6683956	1.4661839	1.3308784	1.393717	1.8618688	2.7133256	3.2947089	1.7213036	1.4730438	1.4709102	1.4465162	2.0867886	3.1920034	3.4556953
6	HOBBIES_1_005_CA_1_validation	0.952025909	0.8484017	0.8553132	0.9223004	1.0622897	1.4702862	1.5094156	0.9749206	0.9690644	0.9741457	0.9098207	1.0017599	1.444084	1.0544859	0.9857176	1.0492929	0.9953305	1.0637917	1.0394462	1.6213455	1.622866	1.0890109	0.8773967	0.9776226	0.9456205	1.1199727	1.5975555	1.5402394
7	HOBBIES_1_006_CA_1_validation	0.904185601	0.8828434	0.7873555	0.8915951	0.8203014	1.117091	1.1270907	0.856187	0.8760622	0.8705127	0.8401848	0.9050329	1.0380628	0.7624651	0.8994553	0.8259233	0.8088896	0.7361009	0.7599577	0.9507622	0.9307187	0.8017357	0.7634047	0.7766329	0.7676251	0.7900375	0.9039065	0.9453005
8	HOBBIES_1_007_CA_1_validation	0.29440125	0.2789668	0.2729298	0.2740767	0.3110941	0.3576879	0.4620524	0.3179739	0.3329747	0.2953394	0.2907108	0.3177043	0.3430288	0.3342962	0.2984038	0.3024227	0.2759485	0.2881225	0.3428825	0.4200767	0.4677394	0.3627898	0.3450986	0.3142096	0.3102786	0.328978	0.4049523	0.4393288
9	HOBBIES_1_008_CA_1_validation	6.755909253	7.5837435	7.7208206	7.1005302	8.3425218	8.6972904	7.6955339	8.61968	8.4939489	7.6064545	8.1787269	8.1741981	8.9760238	5.2635355	8.5922609	9.0202123	8.5194376	7.6860745	8.0252461	8.4370661	8.4781772	8.5364513	7.7599791	7.665133	8.1918859	8.0337205	8.4025779	7.1086413
10	HOBBIES_1_009_CA_1_validation	0.753661417	0.8399959	0.7331377	0.7220257	0.9317001	1.159956	1.204512	0.8594979	0.8236184	0.7512195	0.7179408	0.7704758	1.1650395	1.0150593	0.8549876	0.8268527	0.7400485	0.737672	0.7484596	1.0363275	1.09812	0.8246395	0.753379	0.7833763	0.7706824	0.7679995	1.0874308	1.0836391
11	HOBBIES_1_010_CA_1_validation	0.67460366	0.5239265	0.517798	0.4654875	0.5665589	0.8821112	0.9002093	0.5591366	0.5818709	0.5691602	0.5069401	0.5787958	0.7336144	0.7235062	0.5781739	0.5530198	0.5136887	0.522932	0.5630485	0.8716332	0.9317601	0.5817173	0.5183481	0.5558037	0.5257083	0.6637885	0.8022224	0.8213327
12	HOBBIES_1_011_CA_1_validation	0.075176129	0.0743522	0.071827	0.0747294	0.0922542	0.1080469	0.1352603	0.0867039	0.0819974	0.0749205	0.0814648	0.0906064	0.1217995	0.1224684	0.0843409	0.0822823	0.0768615	0.082924	0.1040273	0.1460902	0.1613924	0.0944324	0.0875331	0.0970404	0.1022903	0.1261118	0.1701599	0.1692718
13	HOBBIES_1_012_CA_1_validation	0.163743782	0.1652771	0.1570569	0.1704209	0.1998576	0.297534	0.3428862	0.1926228	0.1816307	0.1618005	0.1634863	0.1854162	0.262652	0.270449	0.1828215	0.1672062	0.1568121	0.1617598	0.2026675	0.2914623	0.3376935	0.1978069	0.166176	0.1677342	0.1708211	0.1993283	0.2933914	0.3099987
14	HOBBIES_1_013_CA_1_validation	0.319250976	0.2718948	0.2878549	0.2853594	0.3655429	0.4917067	0.5024387	0.3202811	0.3271907	0.3031351	0.3416435	0.3544149	0.4955476	0.418655	0.3052524	0.3010857	0.2848065	0.2875616	0.3781369	0.5639015	0.5866816	0.3616172	0.3572875	0.3585219	0.3512372	0.4482858	0.5938012	0.5928373
15	HOBBIES_1_014_CA_1_validation	1.721128579	1.6354255	1.4763982	1.4736475	1.7258421	1.9715033	2.4004046	1.6714712	1.5928956	1.5483206	1.375502	1.7424828	1.970141	1.6134865	1.5887925	1.5488878	1.4639184	1.566447	1.4930047	1.7503225	1.9391697	1.4968055	1.4919924	1.5085563	1.589487	1.8077325	1.8157657	1.7172109
16	HOBBIES_1_015_CA_1_validation	3.085742894	2.8949252	3.0683983	2.8922999	3.3490913	4.4820514	4.4632555	3.5593612	2.921444	2.8349431	3.0583822	3.6460567	4.6313881	3.1186801	3.014163	3.0163221	2.8825577	2.6820832	3.4892312	4.9426548	4.416523	2.9691775	2.874679	2.8579175	3.0658409	3.9203957	5.1855251	4.8909731
17	HOBBIES_1_016_CA_1_validation	5.468935815	5.2894214	4.5612374	5.0210104	5.5457508	7.79078	6.877531	5.5360481	5.430761	5.2289704	5.0620823	6.2418242	7.0536884	5.025789	4.9820191	6.0244169	5.184761	5.1668022	5.7778122	8.6839549	7.4039897	5.4239763	4.8441774	5.4103524	5.642979	6.2030313	7.4067764	7.0863099
18	HOBBIES_1_017_CA_1_validation	1.006039318	0.8780193	0.8612193	0.8185117	1.1596817	1.5350539	1.941507	1.0036302	0.9688147	0.8865441	0.941258	1.2195442	1.5880533	1.3192661	0.8930291	0.9354637	0.9302984	0.9141636	1.1465393	1.5314276	1.6446302	0.934208	0.9188123	0.9310067	0.91085	1.2769764	1.8687938	1.7253575

- **Final Score;**



Appendix

- **Please check my code:**

https://github.com/claudiawang-tech/M5-Forecasting---Accuracy

- **Reference:**
  - https://www.kaggle.com/rohitsingh9990/m5-forecasting-eda-feature-engineering
  - https://www.kaggle.com/rohitsingh9990/m5-forecasting-eda-feature-engineering