Walmart — Store Sales Forecasting

N

Nishank Dave

5 min read · Apr 27, 2023

For this Machine Learning project, we will use the "Walmart Recruiting — Store Sales Forecasting" dataset, from Kaggle.

link: https://www.kaggle.com/datasets/tanujdhiman/walmart-analysis-dataset?select=myCity.csv

The goal is to predict the Weekly Sales for specific stores, departments and dates. During the flow of this notebook, you will see Exploratory Data Analysis (EDA), Correlation matrix, Ordinary Least Squares (simple linear regression model), Fitting Linear model, Tree Based model, Support Vector

Machine, Multilayer Perceptron Regressor, Used AutoML to find the best model, interpreting SHAP feature importance, and the evaluation metrics which I am using are

- Mean Squared Error
- Root Mean Squared Error
- Mean Absolute Error
- Mean Residual Deviance

Reading the Dataset

```
# Reading Data into Pandas Dataframe
url = 'https://raw.githubusercontent.com/Nishank-NEU/Data-Science/main/walmartSample.csv'
data = pd.read_csv(url, encoding='unicode_escape')  # This code will read the CSV file from the URL provided, and store it in a pandas DataFrame called df.
```

```
data.head(5)
#The head() method is used to display the first 5 rows of the dataset.
```

	Unnamed: 0	X	Store	Date	IsHoliday	Dept	Weekly_Sales	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	Type	Size
0	202017	202017	21	2011-02-25	False	36	1025.00	61.27	3.065	NaN	NaN	NaN	NaN	NaN	213.190421	8.028	B	140167
1	165520	165520	17	2012-03-02	False	31	1236.64	26.28	3.242	14469.08	1163.89	37.38	8771.3	2237.79	130.645793	6.403	B	93188
2	389207	389207	41	2011-12-02	False	94	37971.00	34.53	3.378	4594.56	305.47	1781.24	3168.1	21739.26	195.822329	6.759	A	196321
3	133193	133193	14	2011-05-06	False	67	26400.02	58.21	4.046	NaN	NaN	NaN	NaN	NaN	185.937438	8.521	A	200898
4	401356	401356	43	2010-10-08	False	1	8748.54	72.81	2.633	NaN	NaN	NaN	NaN	NaN	203.605179	10.210	C	41062

Variable Description

1. **Store** — the store number
2. **Dept** — the department number
3. **Date** — the week
4. **Weekly_Sales** — sales for the given department in the given store

5. **IsHoliday** — whether the week is a special holiday week
6. **Temperature** — average temperature in the region.
7. **Fuel_Price** — cost of fuel in the region.
8. **MarkDown1–5** — anonymized data related to promotional markdowns that Walmart is running.
9. **CPI** — the consumer price index.
10. **Unemployment** — the unemployment rate.

Note:- MarkDown data is only available after Nov 2011, and is not available for all stores all the time. Any missing value is marked with an NA.

The Target variable is Weekly_Sales

Exploratory Data Analysis (EDA)

There are some null values, let's review them

```
# Checking Data Type of each variable
data.info()
numeric_data = data.select_dtypes(include=[np.number])
categorical_data = data.select_dtypes(exclude=[np.number])
numeric_data.shape[1]
categorical_data.shape[1]
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 18 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Unnamed: 0    500 non-null    int64
 1   X             500 non-null    int64
 2   Store         500 non-null    int64
 3   Date          500 non-null    object
 4   IsHoliday     500 non-null    bool
 5   Dept          500 non-null    int64
 6   Weekly_Sales  500 non-null    float64
 7   Temperature   500 non-null    float64
 8   Fuel_Price    500 non-null    float64
 9   MarkDown1     170 non-null    float64
 10  MarkDown2     125 non-null    float64
 11  MarkDown3     155 non-null    float64
 12  MarkDown4     154 non-null    float64
 13  MarkDown5     172 non-null    float64
 14  CPI           500 non-null    float64
 15  Unemployment  500 non-null    float64
 16  Type          500 non-null    object
 17  Size          500 non-null    int64
dtypes: bool(1), float64(10), int64(5), object(2)
memory usage: 67.0+ KB
3
```

Data Manipulation

Now, we will do the following steps:

- Remove null values from the markdown variables.
- Create variables for year, month and week, based on the date field.

- Remove the variables with low correlation.

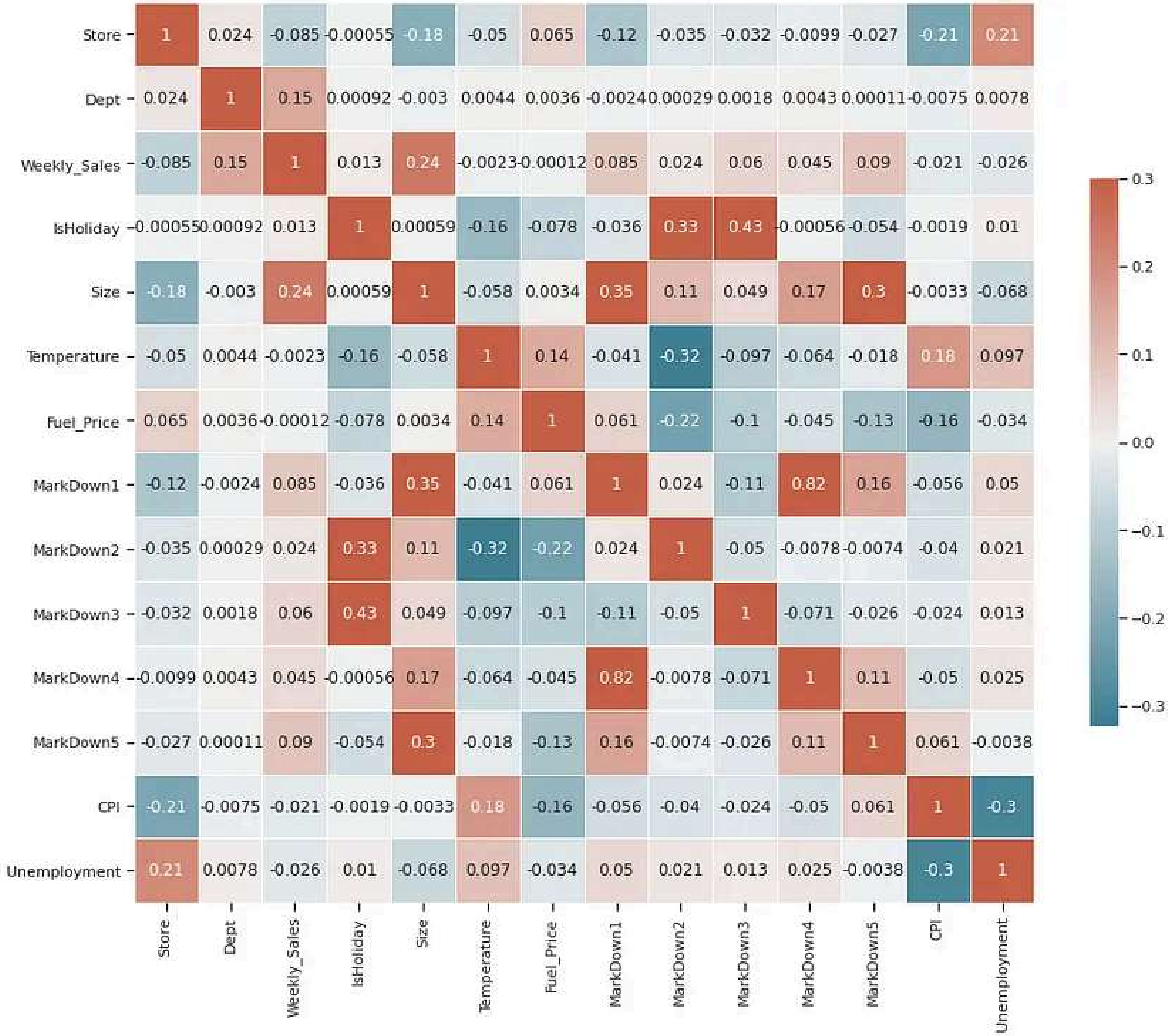
```
[ ] from sklearn.impute import SimpleImputer
    from sklearn.preprocessing import MinMaxScaler, OneHotEncoder
    from sklearn.model_selection import train_test_split
```

```
[ ] data[['MarkDown1','MarkDown2','MarkDown3','MarkDown4', 'MarkDown5']] = data[['MarkDown1','MarkDown2','MarkDown3','MarkDown4','MarkDown5']].fillna(0)
    data['Year'] = pd.to_datetime(data['Date']).dt.year
    data['Month'] = pd.to_datetime(data['Date']).dt.month
    data['Week'] = pd.to_datetime(data['Date']).dt.isocalendar().week
    data = data.drop(columns=["Date", "CPI", "Fuel_Price", 'Unemployment', 'Temperature'])
```

```
[ ] df = data.pop('Weekly_Sales')
    data['Weekly_Sales'] = df
    data
```

	Unnamed: 0	X	Store	IsHoliday	Dept	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	Type	Size	Year	Month	Week	Weekly_Sales
0	202017	202017	21	False	36	0.00	0.00	0.00	0.00	0.00	B	140167	2011	2	8	1025.00
1	165520	165520	17	False	31	14469.08	1163.89	37.38	8771.30	2237.79	B	93188	2012	3	9	1236.64
2	389207	389207	41	False	94	4594.56	305.47	1781.24	3168.10	21739.26	A	196321	2011	12	48	37971.00
3	133193	133193	14	False	67	0.00	0.00	0.00	0.00	0.00	A	200898	2011	5	18	26400.02
4	401356	401356	43	False	1	0.00	0.00	0.00	0.00	0.00	C	41062	2010	10	40	8748.54
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Input Variables Correlation with the output feature Weekly_Sales



Feature Importance and Selection

Lets fit a very simple linear model to understand how the features of walmart dataset is affecting by weekly sales

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           Weekly_Sales   R-squared:                       0.096
Model:                            OLS   Adj. R-squared:                  0.087
Method:                 Least Squares   F-statistic:                     10.55
Date:                Mon, 17 Apr 2023   Prob (F-statistic):           1.23e-09
Time:                        23:38:25   Log-Likelihood:                 464.25
No. Observations:                 500   AIC:                            -916.5
Df Residuals:                     494   BIC:                            -891.2
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.0199      0.016      1.237      0.217      -0.012       0.051
Store         -0.0273      0.015     -1.835      0.067      -0.057       0.002
Dept           0.0279      0.014      1.987      0.047       0.000       0.055
Size           0.0739      0.013      5.546      0.000       0.048       0.100
Year          -0.0131      0.011     -1.180      0.239      -0.035       0.009
Month          0.0329      0.016      2.079      0.038       0.002       0.064
==============================================================================
Omnibus:                      393.498   Durbin-Watson:                   1.939
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             8451.641
Skew:                           3.264   Prob(JB):                         0.00
Kurtosis:                      22.054   Cond. No.                         7.45
==============================================================================
```

Fitting a Linear Model

```
[ ] import sklearn

    linear_model = sklearn.linear_model.LinearRegression()  # Initializing a Linear Model
    linear_model.fit(x_train, y_train)  # Training a linear model
```

```
▾ LinearRegression
LinearRegression()
```

```
[ ] y_linear_predictions = linear_model.predict(x_test).round()
```

Fitting a Tree Based Model

```
[ ] from sklearn.ensemble import RandomForestRegressor

    tree_model = RandomForestRegressor(
        max_depth=X.shape[1], random_state=0, n_estimators=10
    )
    tree_model.fit(x_train, y_train)
```

```
RandomForestRegressor
RandomForestRegressor(max_depth=5, n_estimators=10, random_state=0)
```

```
[ ] y_tree_based_predictions = tree_model.predict(x_test).round()
```

Fitting a Support Vector Machine (SVM)

```
from sklearn import svm

regr = svm.SVR()
svm_model = regr.fit(x_train, y_train)
svm_predictions = svm_model.predict(x_test).round()
```

Fitting a Multilayer Perceptron Regressor model

```
[ ] from sklearn.neural_network import MLPRegressor

    regr = MLPRegressor(random_state=1, max_iter=500).fit(x_train, y_train)
```

```
[ ] mlp_predictions = regr.predict(x_test).round()
```

Using AutoML to find out the best Model

```
autoML[2].leaderboard  # Leaderbord of AutoML output
```

model_id	rmse	mse	mae	rmsle	mean_residual_deviance
StackedEnsemble_BestOfFamily_4_AutoML_1_20230417_233844	0.0655031	0.00429065	0.0421555	0.0562844	0.00429065
GBM_grid_1_AutoML_1_20230417_233844_model_105	0.0678063	0.0045977	0.0445374	0.0582572	0.0045977
StackedEnsemble_AllModels_3_AutoML_1_20230417_233844	0.0682015	0.00465145	0.042541	0.0581336	0.00465145
StackedEnsemble_AllModels_4_AutoML_1_20230417_233844	0.0682649	0.0046601	0.0425002	0.058227	0.0046601
XGBoost_lr_search_selection_AutoML_1_20230417_233844_select_grid_model_2	0.0686443	0.00471204	0.0427949	0.0590538	0.00471204
GBM_grid_1_AutoML_1_20230417_233844_model_102	0.0688452	0.00473967	0.0446187	0.059105	0.00473967
GBM_grid_1_AutoML_1_20230417_233844_model_41	0.0696758	0.00485471	0.0457239	0.0598178	0.00485471
GBM_grid_1_AutoML_1_20230417_233844_model_93	0.0705054	0.00497102	0.0467537	0.0607115	0.00497102
StackedEnsemble_AllModels_2_AutoML_1_20230417_233844	0.0705227	0.00497345	0.0447089	0.0604447	0.00497345
GBM_grid_1_AutoML_1_20230417_233844_model_12	0.0706145	0.0049864	0.0444409	0.0603462	0.0049864

[184 rows x 6 columns]

The evaluation metrics which I am using are

- Mean Squared Error
- Root Mean Squared Error
- Mean Absolute Error
- Mean Residual Deviance

```
Model Details
=============
H2ORandomForestEstimator : Distributed Random Forest
Model Key: gbm_grid2_model_49
```

Model Summary:

number_of_trees	number_of_internal_trees	model_size_in_bytes	min_depth	max_depth	mean_depth	min_leaves	max_leaves	mean_leaves
100.0	100.0	139031.0	10.0	10.0	10.0	83.0	130.0	106.08

```
ModelMetricsRegression: drf
** Reported on train data. **

MSE: 0.006465710740288608
RMSE: 0.08040964332894786
MAE: 0.05389729492753286
RMSLE: 0.06925425571759158
Mean Residual Deviance: 0.006465710740288608

ModelMetricsRegression: drf
** Reported on validation data. **

MSE: 0.01742398926900791
RMSE: 0.13199995935229644
MAE: 0.0709195756845966
RMSLE: 0.10229989309160167
Mean Residual Deviance: 0.01742398926900791
```

Scoring History:

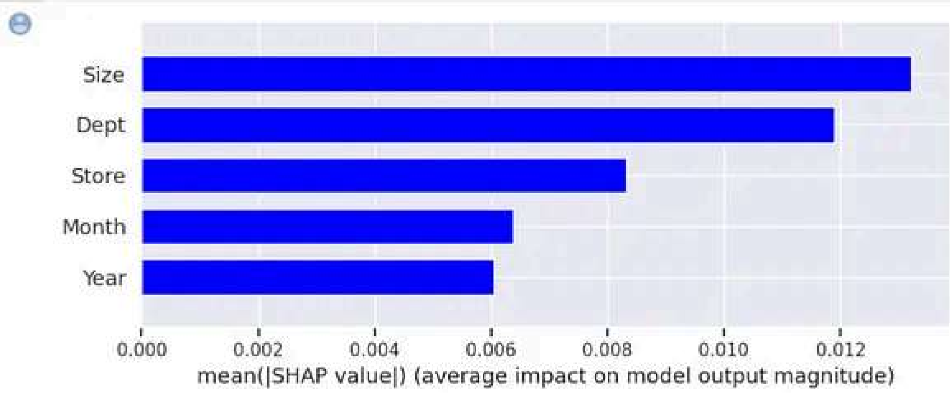
	timestamp	duration	number_of_trees	training_rmse	training_mae	training_deviance	validation_rmse	validation_mae	validation_deviance
	2023-04-17 23:42:57	13.006 sec	0.0	nan	nan	nan	nan	nan	nan
	2023-04-17 23:42:57	13.008 sec	1.0	0.1294405	0.0794224	0.0167548	0.1517974	0.0792146	0.0230424
	2023-04-17 23:42:57	13.011 sec	2.0	0.1212211	0.0758477	0.0146946	0.1336395	0.0654572	0.0178595
	2023-04-17 23:42:57	13.013 sec	3.0	0.1113435	0.0712494	0.0123974	0.1288159	0.0658559	0.0165935
	2023-04-17 23:42:57	13.015 sec	4.0	0.1050954	0.0695462	0.0110450	0.1233157	0.0645316	0.0152068
	2023-04-17 23:42:57	13.017 sec	5.0	0.1006247	0.0663492	0.0101253	0.1255268	0.0667667	0.0157570
	2023-04-17 23:42:57	13.019 sec	6.0	0.0978834	0.0638965	0.0095812	0.1261144	0.0669745	0.0159048
	2023-04-17 23:42:57	13.021 sec	7.0	0.0934346	0.0613169	0.0087300	0.1251468	0.0659609	0.0156617
	2023-04-17 23:42:57	13.023 sec	8.0	0.0907706	0.0595502	0.0082393	0.1269447	0.0687905	0.0161149
	2023-04-17 23:42:57	13.026 sec	9.0	0.0900269	0.0582537	0.0081048	0.1288142	0.0703404	0.0165931
---	---	---	---	---	---	---	---	---	---
	2023-04-17 23:42:57	13.375 sec	91.0	0.0806163	0.0540263	0.0064990	0.1316523	0.0709474	0.0173323
	2023-04-17 23:42:57	13.381 sec	92.0	0.0804624	0.0540410	0.0064742	0.1316679	0.0708599	0.0173364
	2023-04-17 23:42:57	13.387 sec	93.0	0.0805523	0.0541223	0.0064887	0.1317379	0.0709037	0.0173549
	2023-04-17 23:42:57	13.393 sec	94.0	0.0804053	0.0539930	0.0064650	0.1317152	0.0708590	0.0173489
	2023-04-17 23:42:57	13.400 sec	95.0	0.0804418	0.0540841	0.0064709	0.1318702	0.0709765	0.0173898
	2023-04-17 23:42:57	13.406 sec	96.0	0.0803955	0.0540131	0.0064634	0.1317594	0.0709471	0.0173605
	2023-04-17 23:42:57	13.412 sec	97.0	0.0803401	0.0538797	0.0064545	0.1319404	0.0711002	0.0174083
	2023-04-17 23:42:57	13.419 sec	98.0	0.0803555	0.0538525	0.0064570	0.1320001	0.0711034	0.0174240
	2023-04-17 23:42:57	13.480 sec	99.0	0.0802977	0.0538620	0.0064477	0.1320099	0.0709985	0.0174266
	2023-04-17 23:42:57	13.489 sec	100.0	0.0804096	0.0538973	0.0064657	0.1320000	0.0709196	0.0174240

[101 rows x 10 columns]

Interpreting SHAP Feature Importance Plot for Linear and Tree-based model

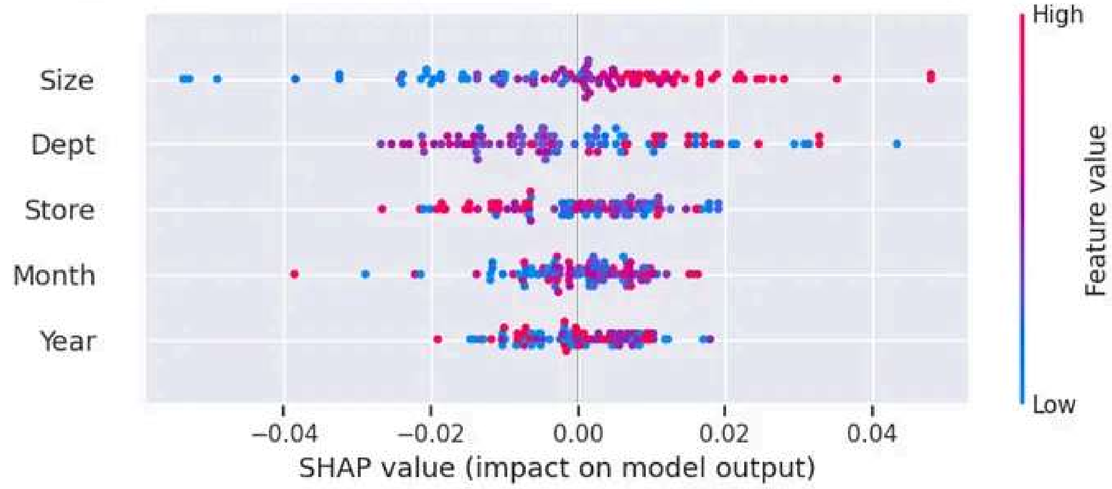
The idea behind SHAP feature importance is simple: Features with large absolute Shapley values are important. Since we want global importance, we average the absolute Shapley values per feature across the data. Next, we sort the features by decreasing importance and plot them.

```
# SHAP variable importance for a Linear Model
shap.summary_plot(svm_shap_values, x_train, plot_type="bar", color="blue")
```

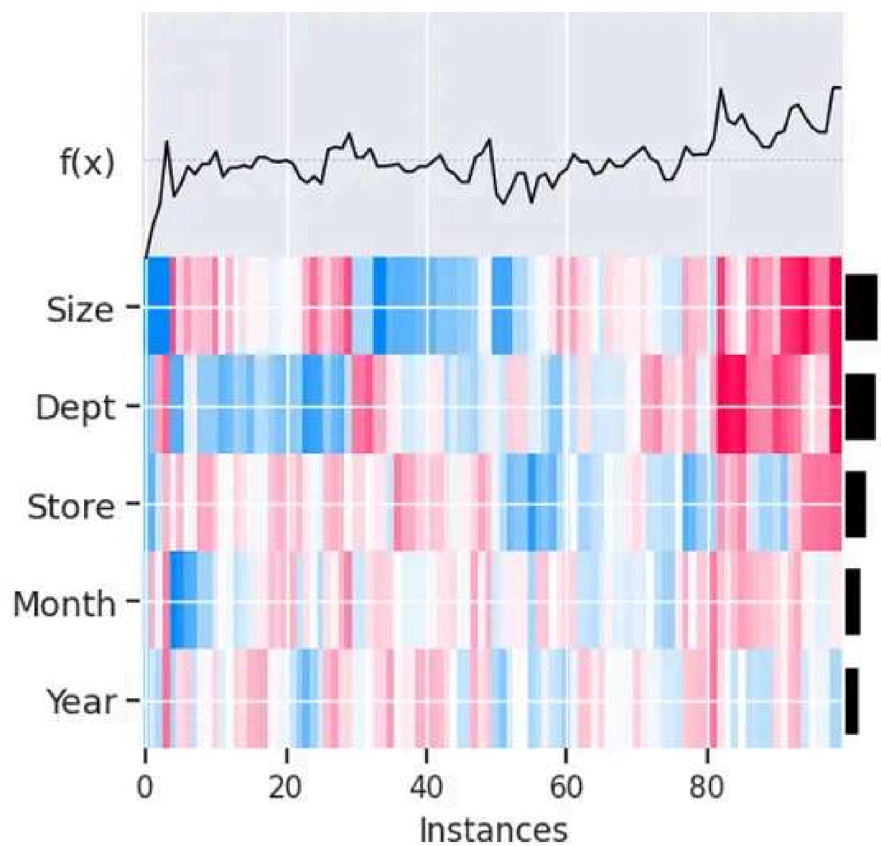


```
[ ] # SHAP summary for Linear Model
    shap.summary_plot(svm_shap_values, x_train_100)
```

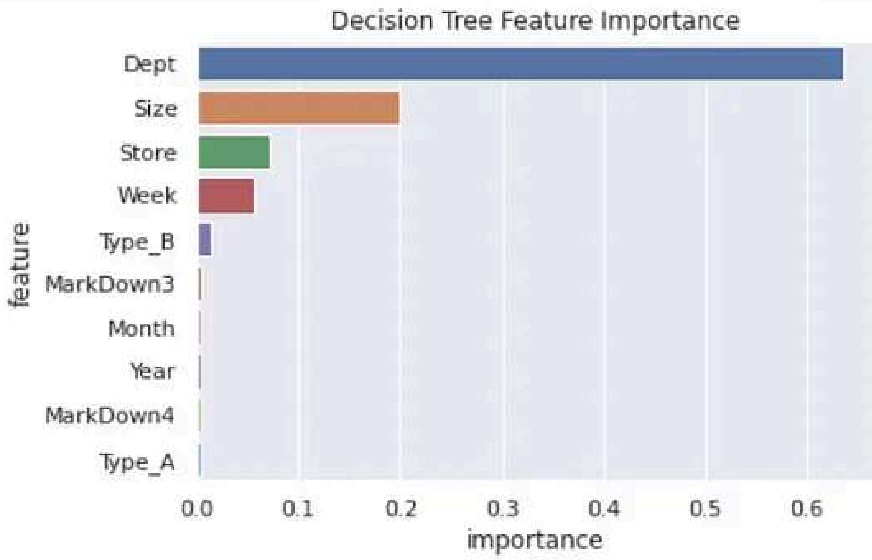
No data for colormapping provided via 'c'. Parameters 'vmin', 'vmax' will be ignored




```
shap.plots.heatmap(svm_shap_values)  # SHAP HeatMap of a Tree Based Model
```

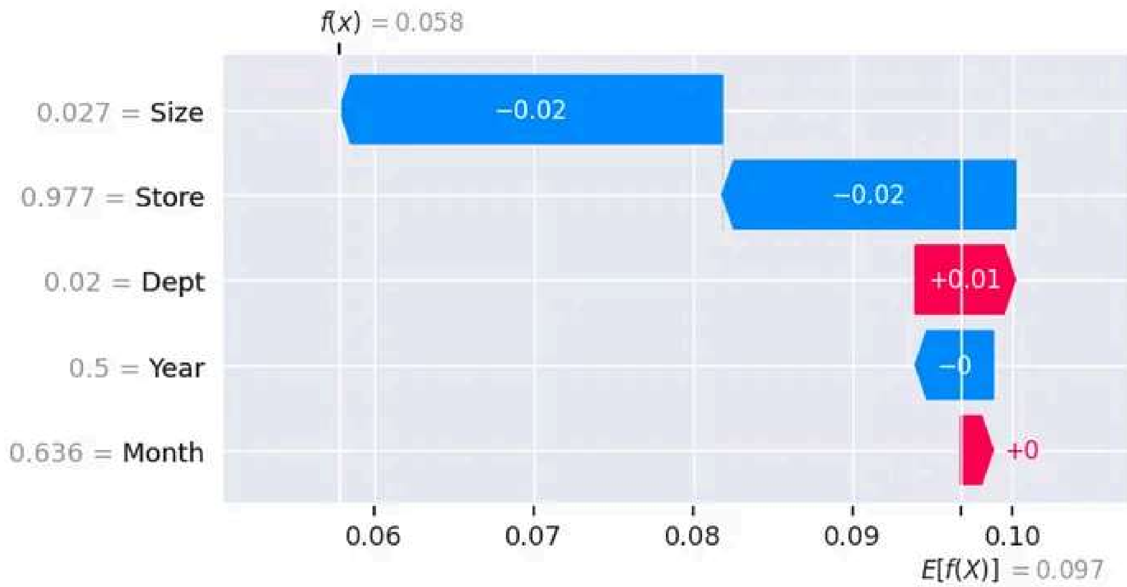


```
plt.title('Decision Tree Feature Importance')
sns.barplot(data=tree_importance_df.head(10), x='importance', y='feature');
```



Interpreting Waterfall SHAP visualization Let's consider the same sample (sample_ind = 18). It says that f(x) = 0.058 is what we got as a model output and the expected output for this sample was 0.097. We came pretty close to determining it as the difference is only 0.039. The waterfall model explains how we got the expected output, and which features contributed to what. The below graph shows that weekly sales has the biggest and most positive impact in increasing the dept by 0.01 for this specific sample. Followed by size had a negative impact and it bought the weekly sales down again by 0.02 for this sample, and so on. Using this model we can visually interpret why exactly this specific sample is giving an output of 0.097

```
[ ] get_SHAP()
```



Learning Outcomes

I learned the complete lifecycle of a Data Science project right from data preparation to hyperparameter tuning Majority of the time should be invested in data preparation i.e. cleaning the data, normalizing, feature selection, imputation etc. Hyperparameter tuning is the second most important thing after data preparation, which most of the practioner's ignore. But the results are worth the time invested Multiple models must be trained and the best models should be selected to be deployed, as some algorithms perform much better than the other's on specific tasks Model Interpretation(Unboxing the Black Box) is the best takeaway from the series of this assignments. SHAP, LIME and PDP have made it easier to understand what made a model to predict a outcome.

References

1. https://docs.h2o.ai/h2o/latest-stable/h2o-docs/grid-search.html
2. https://towardsdatascience.com/3.explain-your-model-with-the-shap-values-bc36aac4de3d

3. https://medium.com/@sergioalves94/walmart-store-sales-forecasting-4ffebbbf650f

4. https://www.kaggle.com/code/avelinocaio/walmart-store-sales-forecasting/notebook

5. https://www.kaggle.com/c/walmart-recruiting-store-sales-forecasting

Authors

1. Nishank Dave (https://www.linkedin.com/in/nishankdave/)

2. Prof. Nik Bear Brown (https://medium.com/@NikBearBrown)

Dall-E

DALL-E is a powerful AI-based image generation tool that can be used to create unique and stunning artwork for your Medium articles or...

4 min read · Apr 21, 2023

See all from Nishank Dave

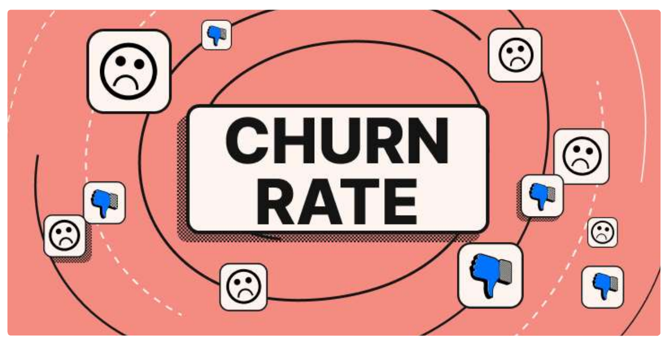
Recommended from Medium

Kai Cansler

Customer Churn

Flatiron DSC Phase 3 Project using Classification Model

7 min read · Feb 9, 2024



Nicholas Aprilie

E-Commerce Customer Churn Prediction

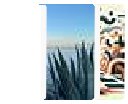
The e-commerce sector has witnessed remarkable growth in recent years, marked ...

7 min read · Nov 27, 2023

56

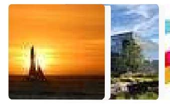
50

Lists



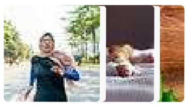
Staff Picks

597 stories · 807 saves



Stories to Help You Level-Up at Work

19 stories · 513 saves

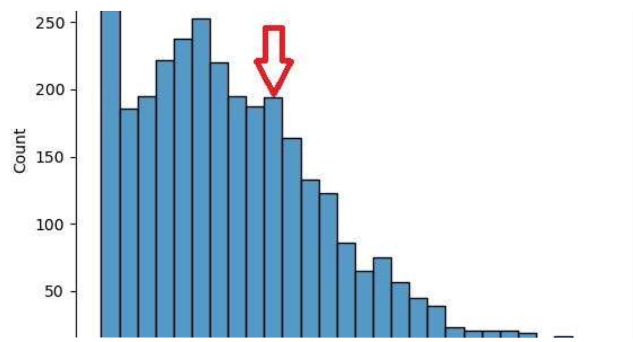


Self-Improvement 101

20 stories · 1461 saves

Productivity 101

20 stories · 1344 saves



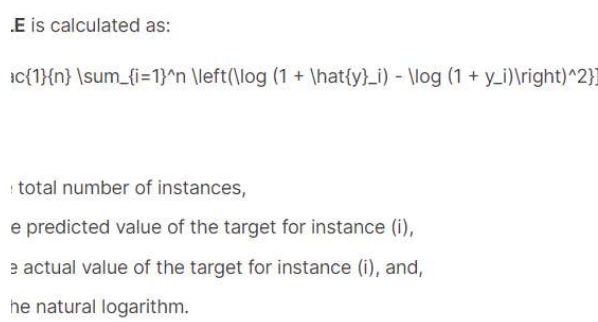
Sohrab Salehin in Stackademic

How to label churn customers in non-contractual businesses

If you are a data analyst, you probably asked to calculate churn rate for your employer…

4 min read · Nov 15, 2023

192

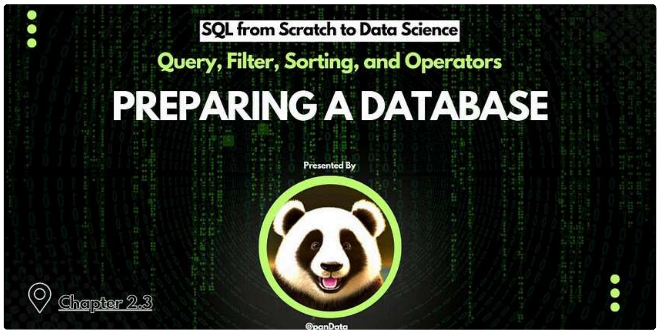


Pratikpophali

Grocery Sales Prediction Using Time Series Forecasting ML Model

Forecasts aren't just for meteorologists. Governments forecast economic growth.…

6 min read · Sep 24, 2023



panData in Level Up Coding

SQL Series: Preparing a Database Step by Step

SQL from Scratch to Data Science [2.2]

12 min read · 5 days ago

141

Anantharaman Janakir... in Expedia Group Techn...

Expedia Group's Customer Lifetime Value Prediction Model

Building for profitability: Why we created and implemented a CLV model

11 min read · Sep 14, 2023

150 1

See more recommendations