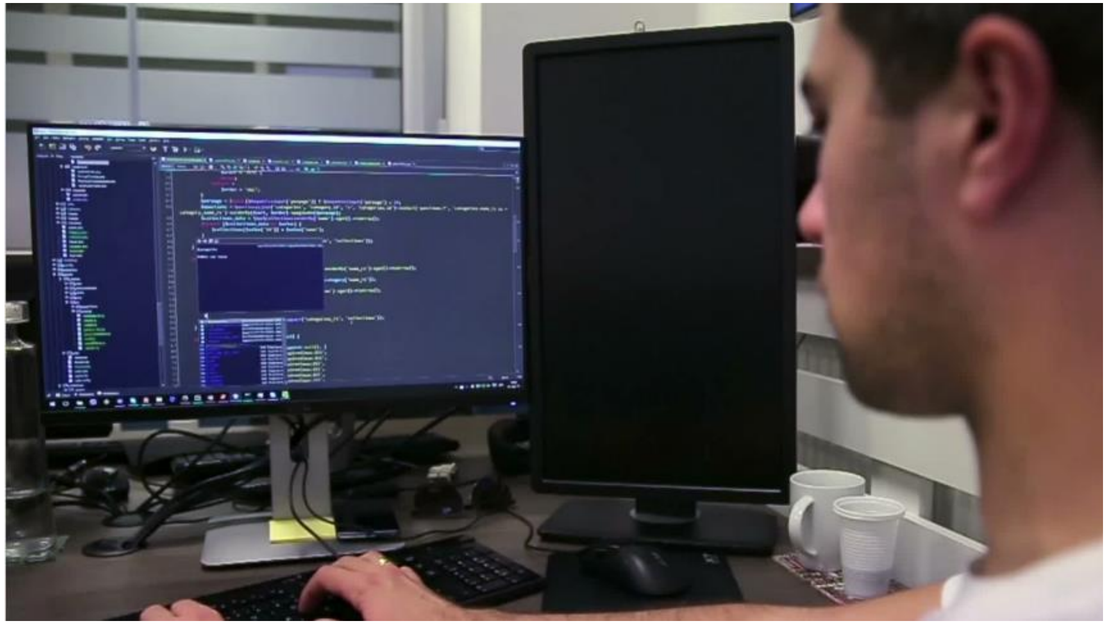
DS105 PROJECT OUTCOME ON LOGISTICS INDUSTRY

Presented By: Daniel Tan

CONTENTS

- Current Problem
- Aim, Rationale and chosen Machine Learning model
- First glance of the Dataset
- Exploratory Data Analysis
- Regression ML Training & Prediction
  - Dummy & Scaling
  - PyCaret (Regression)
  - Gradient Boosting Regressor (Predictions)
- Classification ML Training & Prediction
  - Scaling & SMOTE
  - PyCaret (Classification)
  - Light Gradient Boosting Machine (Predictions)
- Conclusion
- Limitations and Takeaways



CURRENT PROBLEM

A Logistics company is trying to determine which products to continue selling, and which products to remove from their inventory.

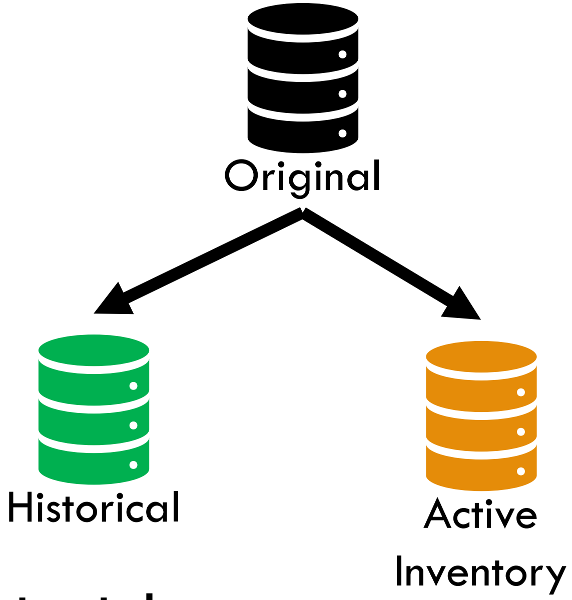
AIM, RATIONALE AND CHOSEN MACHINE LEARNING (ML) MODEL

Aim: Determine which product(s) to continue stocking based on the sales within the past 6 months.

Rationale: Reason is to maximise the spatial value of the inventory warehouse.

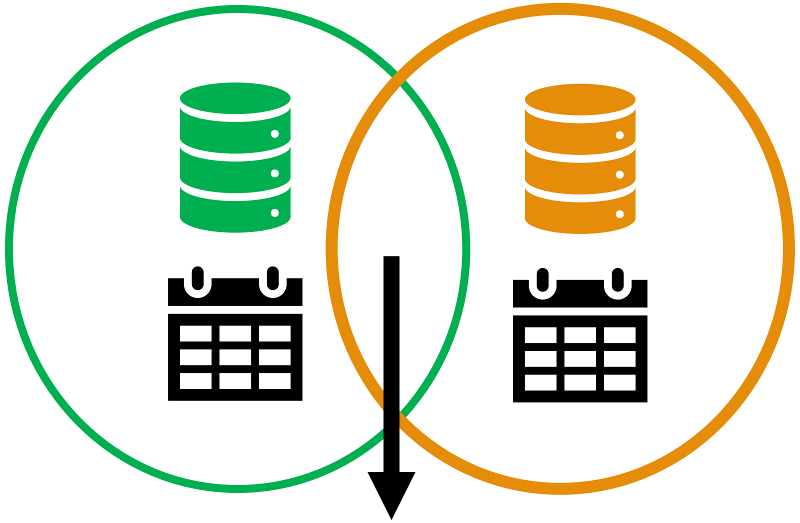
Chosen ML model: Supervised Regressor

FIRST GLANCE OF THE DATASET



Dataset shapes:

- Original dataset = 198917 rows, 14 columns
- Historical dataset = 75996 rows, 14 columns
- Active Inventory dataset = 122921 rows, 14 columns

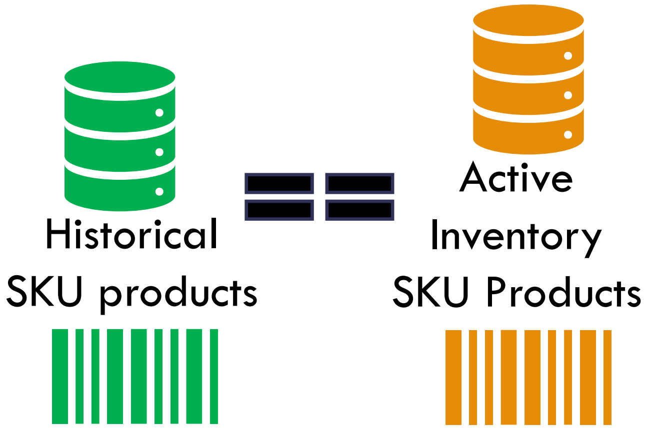


Common overlapping years were 2000 – 2017 after listing them out.

Products release year:

- Historical dataset = 1935 to 2017*
- Active Inventory = 1900 to 2018*

*Caveat: Not in consecutive order



Use unique SKU Numbers from both datasets for train and testing of ML model.

Dataset shape (To Be Pre-processed using merge inner join)

EXPLORATORY DATA ANALYSIS

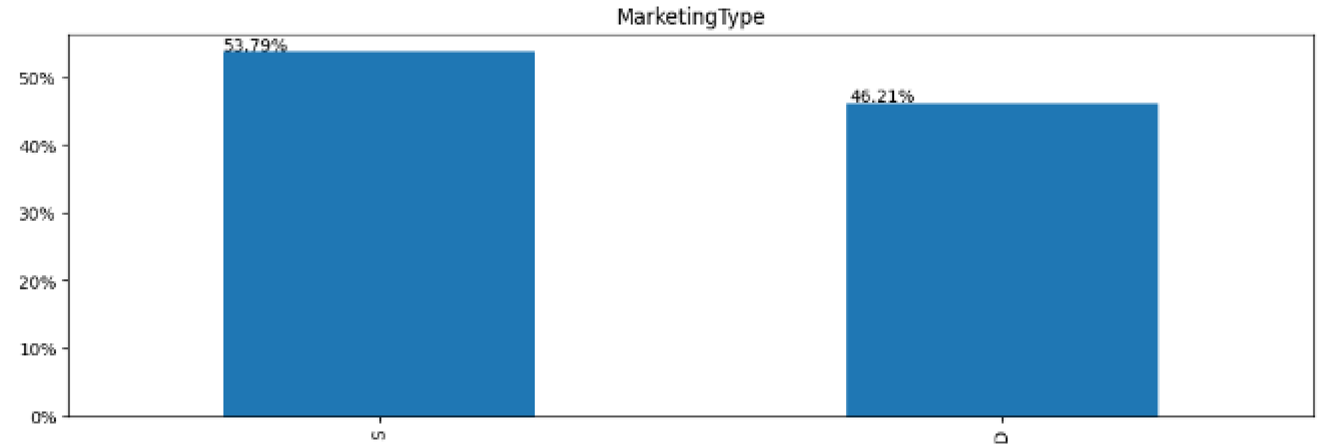
Features	
SKU_number	Unique identifier for each product (198 917 products)
PriceReg	Product Price
File_Type	Historical sales or Active Inventory
Release_Year	Year which Product release
ItemCount	Quantity of Items left in Inventory
MarketingType	S = Smarketing (method that fuses sales and marketing) D = Direct marketing (e.g. marketing towards end consumers)
ReleaseNumber	Number of version/iterations the product has had

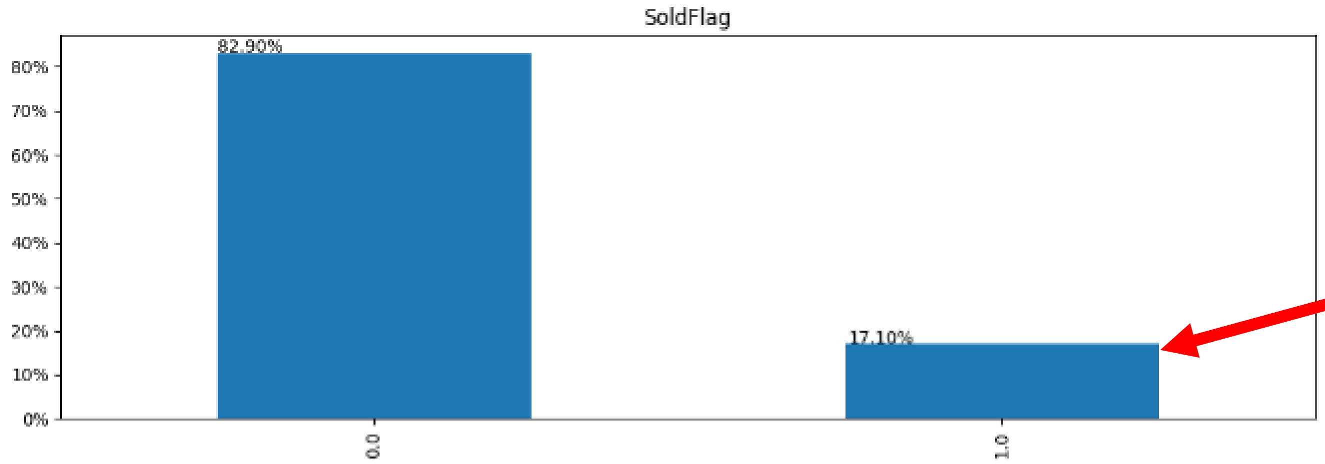
Labels	
Sold_Count	Units sold for that product
SoldFlag	1 = product sold within 6 months 0 = product was not sold within 6 months

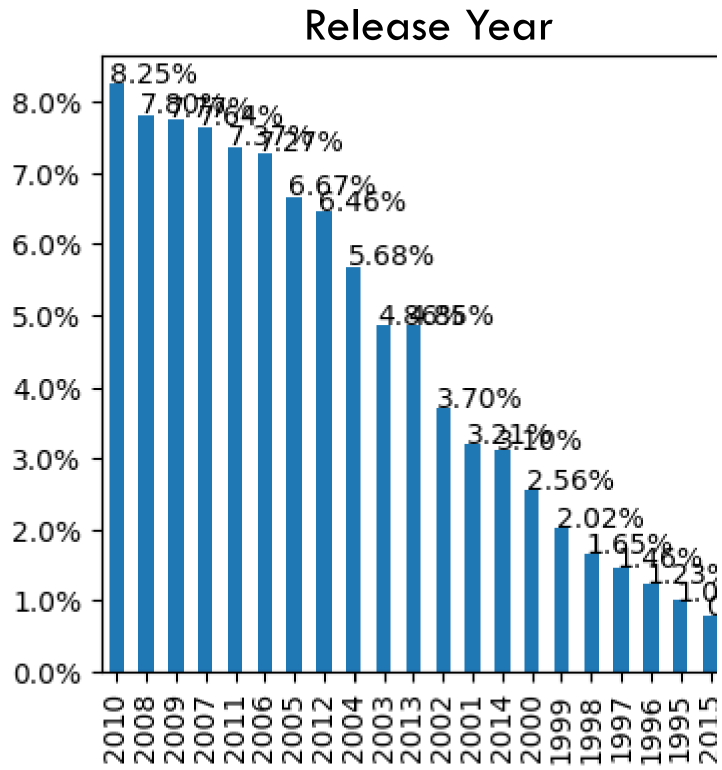
Data.shape = (198917, 14)

**Discarded features: "LowUserPrice", "LowNetPrice" and "StrengthFactor"*

EXPLORATORY DATA ANALYSIS (CATEGORICAL DATA)

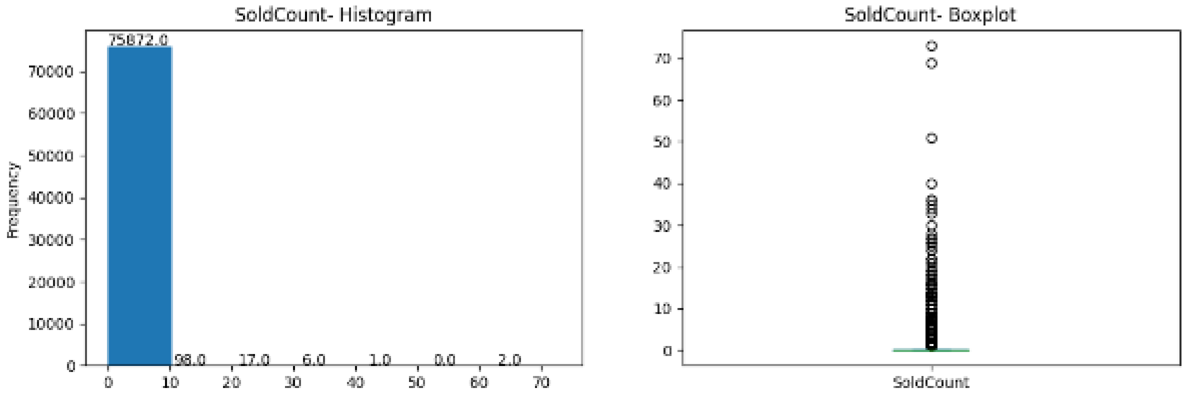


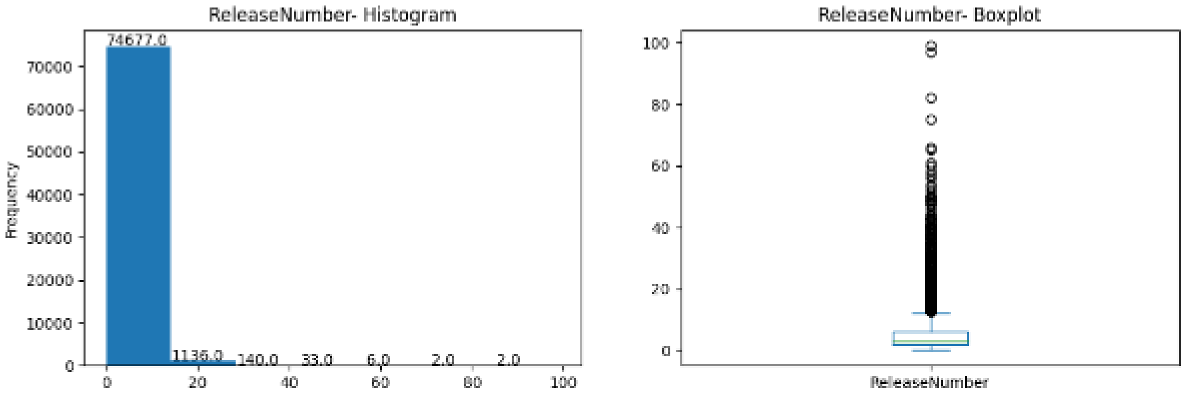


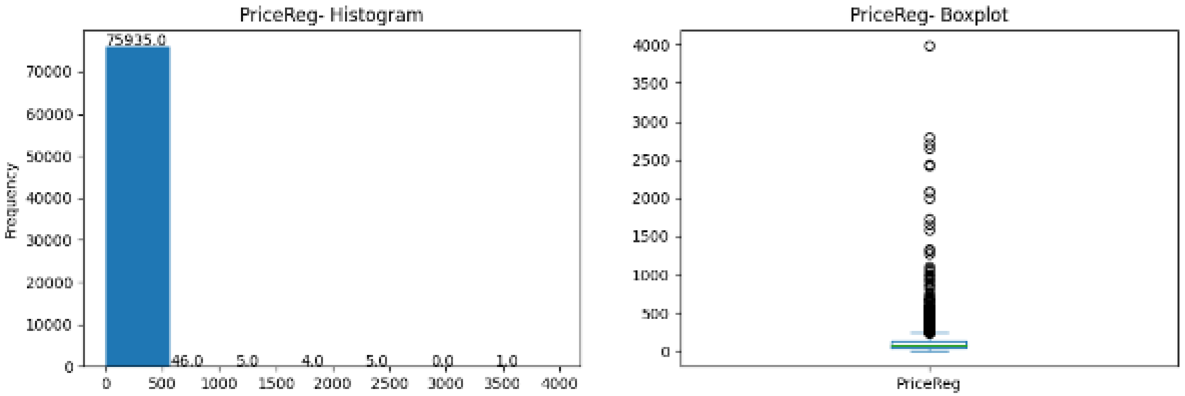


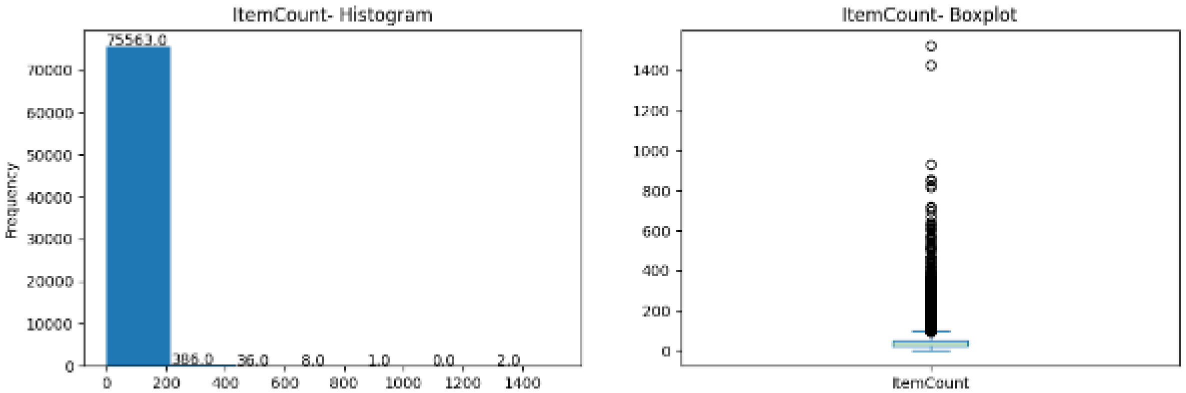
Severely imbalanced

EXPLORATORY DATA ANALYSIS (NUMERICAL DATA)









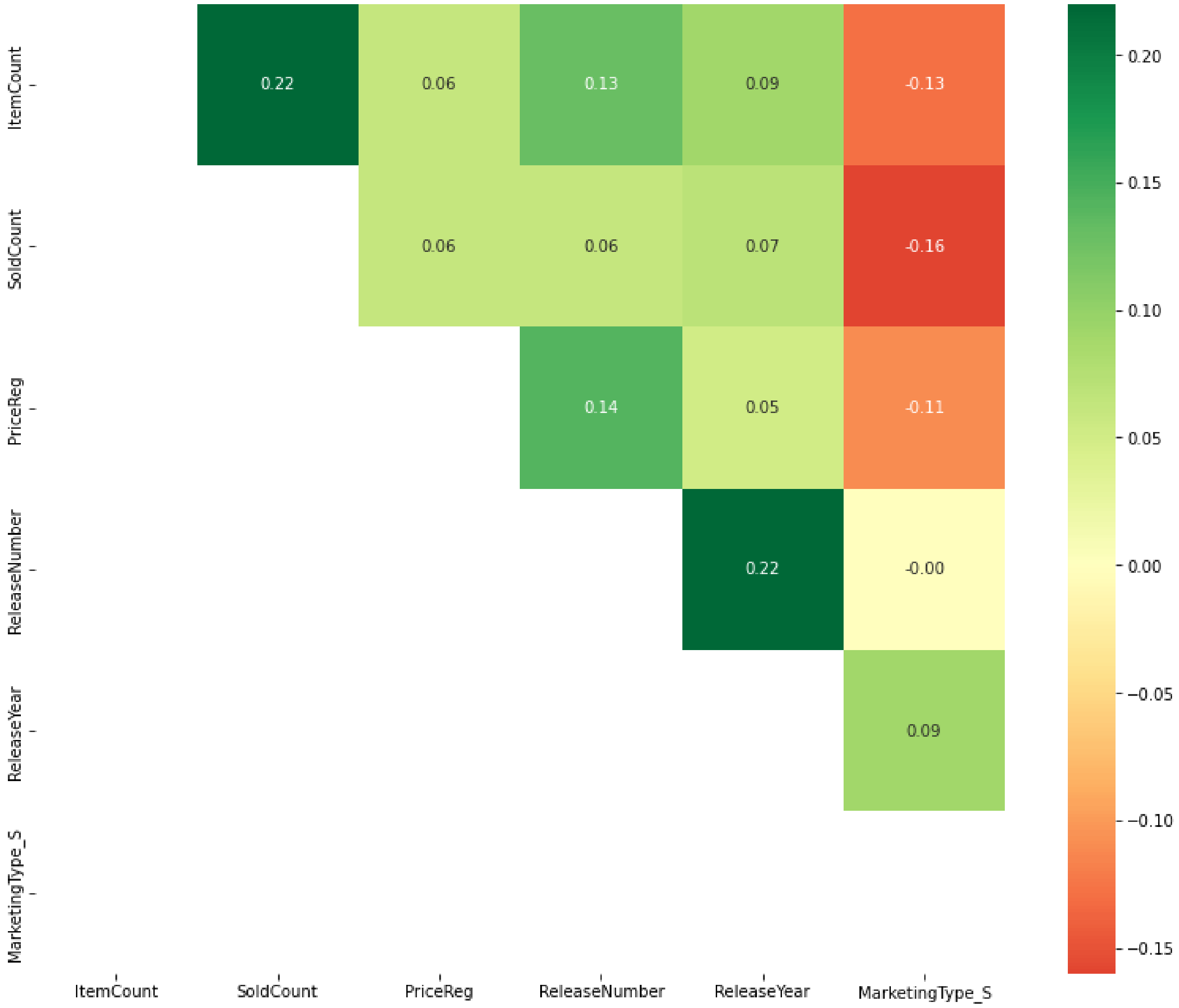
EXPLORATORY DATA ANALYSIS (CORRELATION HEATMAP)

Positive Correlation Features:

- Item count ⇔ Sold Count
- Release Year ⇔ Release Number
- Item count ⇔ Release Number
- Price Reg ⇔ Release Number

Negative Correlational insights:

- S-Marketing* ⇔ Sold Count
- S-Marketing ⇔ Item Count



REGRESSION ML TRAINING & PREDICTION

DUMMY & SCALING FOR REGRESSOR ML TRAINING

- Dummy variables

```
data_historical_SKU = pd.get_dummies(data_historical_SKU, drop_first=True)

data_historical_SKU

# Marketing type = 0, means it is direct marketing
# Marketing type = 1, means it is a integrated marketing type with sales and marketing departments
```

✓ 0.6s

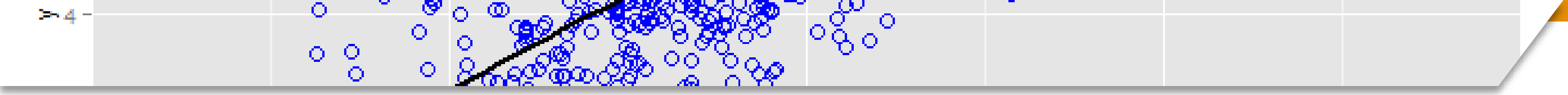
	SoldCount_hist	ReleaseNumber_hist	PriceReg_hist	ReleaseYear_hist	MarketingType_hist_S
0	0.0	10	0.00	2015	0
1	1.0	2	599.00	2012	0
2	3.0	4	0.00	2012	0
3	1.0	4	104.75	2012	0
4	3.0	13	292.00	2013	0
...	...	...	...	...	...
65552	0.0	2	64.99	2008	1
65553	0.0	6	21.50	2004	1
65554	0.0	7	142.75	2006	1
65555	0.0	2	50.00	2012	1
65556	0.0	2	46.95	2001	1

- Scaling

```
#Scaling

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(data_historical_SKU)

col_name = data_historical_SKU.columns

data_historical_SKU = scaler.transform(data_historical_SKU)
data_historical_SKU = pd.DataFrame(data_historical_SKU, columns=col_name)
data_historical_SKU = data_historical_SKU.join(SoldCount_hist)

data_historical_SKU
```

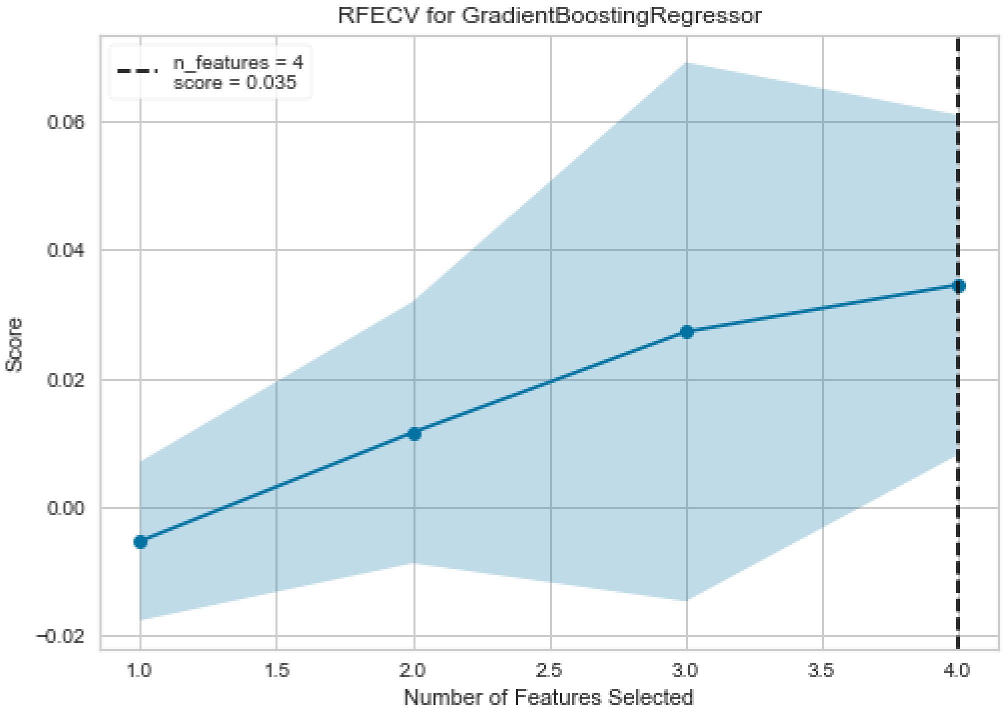
✓ 0.3s

	ReleaseNumber_hist	PriceReg_hist	ReleaseYear_hist	MarketingType_hist_S	SoldCount_hist
0	1.522531	-1.279978	1.509194	-1.041608	0.0
1	-0.536564	6.526896	0.999633	-1.041608	1.0
2	-0.021790	-1.279978	0.999633	-1.041608	3.0
3	-0.021790	0.085248	0.999633	-1.041608	1.0
4	2.294691	2.525710	1.169487	-1.041608	3.0



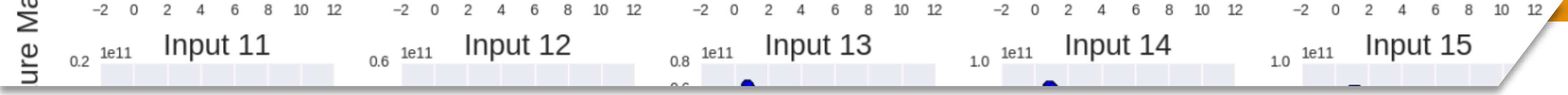
PYCARET (REGRESSION)

- Compare_models()
- Evaluate_model(<insert best model>)



	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
gbr	Gradient Boosting Regressor	0.5425	1.6674	1.2802	0.0345	0.4214	0.6365
lar	Least Angle Regression	0.5584	1.6685	1.2813	0.0326	0.4265	0.6323
br	Bayesian Ridge	0.5585	1.6685	1.2813	0.0326	0.4264	0.6325
lr	Linear Regression	0.5584	1.6685	1.2813	0.0326	0.4265	0.6323
ridge	Ridge Regression	0.5584	1.6685	1.2813	0.0326	0.4265	0.6323
lightgbm	Light Gradient Boosting Machine	0.5433	1.6672	1.2810	0.0324	0.4265	0.6550
omp	Orthogonal Matching Pursuit	0.5641	1.6830	1.2869	0.0239	0.4284	0.6500
llar	Lasso Least Angle Regression	0.6005	1.7235	1.3026	-0.0004	0.4383	0.7151
lasso	Lasso Regression	0.6005	1.7235	1.3026	-0.0004	0.4383	0.7151
en	Elastic Net	0.6005	1.7235	1.3026	-0.0004	0.4383	0.7151
dummy	Dummy Regressor	0.6005	1.7235	1.3026	-0.0004	0.4383	0.7151
xgboost	Extreme Gradient Boosting	0.5513	1.8478	1.3462	-0.0678	0.4356	0.6801
huber	Huber Regressor	0.3720	1.8618	1.3550	-0.0847	0.4563	1.0000
knn	K Neighbors Regressor	0.5659	1.9767	1.3981	-0.1597	0.4833	0.7632
rf	Random Forest Regressor	0.5774	2.0006	1.4067	-0.1782	0.4803	0.7444
et	Extra Trees Regressor	0.5880	2.4811	1.5688	-0.4756	0.5166	0.8603
par	Passive Aggressive Regressor	0.9317	2.5581	1.5905	-0.5228	0.6081	0.8592
dt	Decision Tree Regressor	0.6236	3.2198	1.7884	-0.9354	0.5600	0.9419
ada	AdaBoost Regressor	3.0274	18.1527	3.7513	-8.5158	1.2019	1.9654

*Based on the poor R-squared score, and the high RMSE, I am changing from regression to classification;
But will try to tune the hyperparameters and use GB Regressor just to test out how badly it might look.



GRADIENT BOOSTING REGRESSOR (PREDICTIONS)

```python
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import GradientBoostingRegressor

GBR = GradientBoostingRegressor()

param_grid = {
    'alpha': [0.9],
    'ccp_alpha': [0.0],
    'criterion': ['friedman_mse'],
    'learning_rate': [0.05, 0.055],
    'loss': ['ls'],
    'max_depth': [3],
    'min_impurity_decrease': [0.0],
    'min_samples_leaf': [1],
    'min_samples_split': [2],
    'min_weight_fraction_leaf': [0.0],
    'n_estimators': [90],
    'presort': ['deprecated'],
    'random_state': [7801],
    'subsample': [1.0],
    'tol': [0.0001],
    'validation_fraction': [0.1],
    'verbose': [0],
    'warm_start': ['False']
}
```

0.1s

```python
optimal_params = GridSearchCV(
                    estimator = GBR,
                    param_grid = param_grid,
                    scoring = 'r2',
                    verbose = 0,
                    cv = 3
)

optimal_params.fit(features_train,
                   target_train,
)
```

R2 score improved slightly from 0.0345 to 0.0506

Tuned with GridSearchCV

- Learning rate = 0.05
- max_depth = 3
- n_estimators = 90
- random_state = 7801

Predictions (ranked based on Price of product)

SKU_number	MarketingType	ReleaseNumber	PriceReg	ReleaseYear	ItemCount_active	Predicted_SoldCount
3504396	S	8	317.00	2016	0	1.0
3504561	S	5	274.99	2016	0	1.0
3504528	S	7	274.99	2016	0	1.0
3467973	S	2	250.00	2016	0	1.0
2351093	S	5	160.00	2016	0	1.0
3483275	S	2	149.95	2016	0	1.0
1438250	S	8	116.75	2008	0	1.0
858486	S	4	115.00	2015	0	1.0
1440598	S	9	112.00	2011	0	1.0
803858	S	2	89.95	2013	0	1.0

Total of 30 products needs to be restocked based on the GB Regressor prediction.

CLASSIFICATION ML TRAINING & PREDICTION

SCALING & SMOTE FOR CLASSIFICATION ML TRAINING

- Scaling

```
scaler.fit(x_hist)

col_name = x_hist.columns

x_hist = scaler.transform(x_hist)
x_hist = pd.DataFrame(x_hist, columns=col_name)

x_hist
```

✓ 0.4s

	ReleaseNumber_hist	PriceReg_hist	ReleaseYear_hist	MarketingType_hist_S
0	1.522531	-1.279978	1.509194	-1.041608
1	-0.536564	6.526896	0.999633	-1.041608
2	-0.021790	-1.279978	0.999633	-1.041608
3	-0.021790	0.085248	0.999633	-1.041608
4	2.294691	2.525710	1.169487	-1.041608
...	...	...	...	...
65552	-0.536564	-0.432951	0.320218	0.960054
65553	0.492983	-0.999764	-0.359196	0.960054
65554	0.750370	0.580509	-0.019489	0.960054
65555	-0.536564	-0.628319	0.999633	0.960054
65556	-0.536564	-0.668070	-0.868757	0.960054

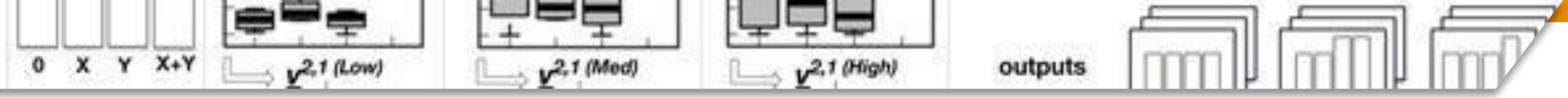
- SMOTE

```
import imblearn
```

✓ 0.1s

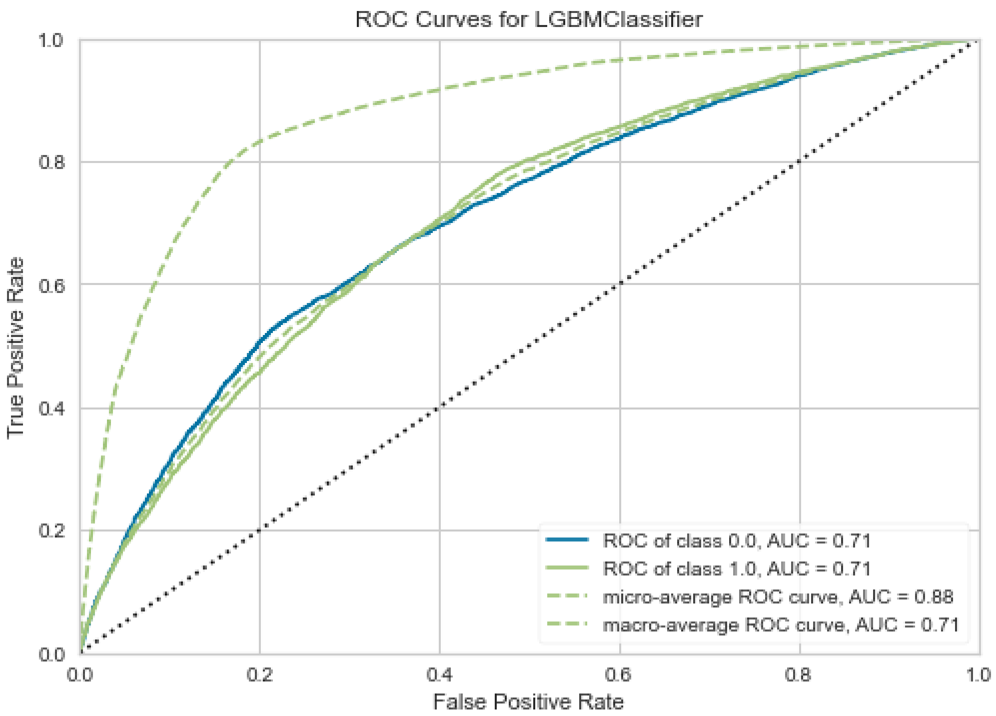
```
sm = imblearn.over_sampling.SMOTE(random_state=42)
x_hist_res, y_hist_res = sm.fit_resample(x_hist, y_hist)
```

✓ 0.1s



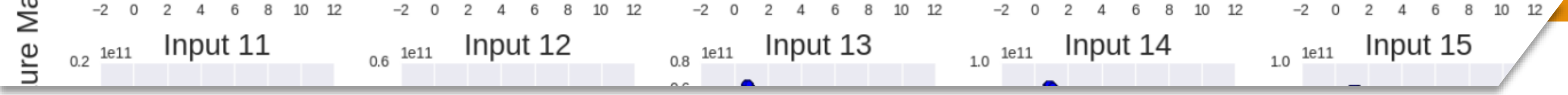
PYCARET (CLASSIFICATION)

- Compare_models()
- Evaluate_model(<insert best model>)



	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
lightgbm	Light Gradient Boosting Machine	0.8086	0.6991	0.0513	0.5479	0.0937	0.0627	0.1217
gbc	Gradient Boosting Classifier	0.8078	0.7027	0.0431	0.5258	0.0796	0.0519	0.1067
ada	Ada Boost Classifier	0.8074	0.6949	0.0167	0.5463	0.0323	0.0210	0.0676
dummy	Dummy Classifier	0.8070	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
ridge	Ridge Classifier	0.8069	0.0000	0.0010	0.4367	0.0020	0.0011	0.0127
lr	Logistic Regression	0.8067	0.6795	0.0041	0.4256	0.0080	0.0045	0.0252
lda	Linear Discriminant Analysis	0.8066	0.6863	0.0082	0.4621	0.0162	0.0094	0.0396
xgboost	Extreme Gradient Boosting	0.8051	0.6948	0.0744	0.4713	0.1283	0.0799	0.1247
qda	Quadratic Discriminant Analysis	0.7957	0.6796	0.0789	0.3658	0.1296	0.0656	0.0910
nb	Naive Bayes	0.7904	0.6774	0.0949	0.3442	0.1487	0.0714	0.0909
knn	K Neighbors Classifier	0.7791	0.5971	0.1313	0.3235	0.1866	0.0845	0.0962
rf	Random Forest Classifier	0.7570	0.6252	0.2206	0.3152	0.2594	0.1196	0.1223
et	Extra Trees Classifier	0.7437	0.6049	0.2413	0.2978	0.2664	0.1133	0.1142
svm	SVM - Linear Kernel	0.7404	0.0000	0.1329	0.2351	0.0889	0.0273	0.0433
dt	Decision Tree Classifier	0.7261	0.5501	0.2516	0.2774	0.2691	0.1009	0.1010

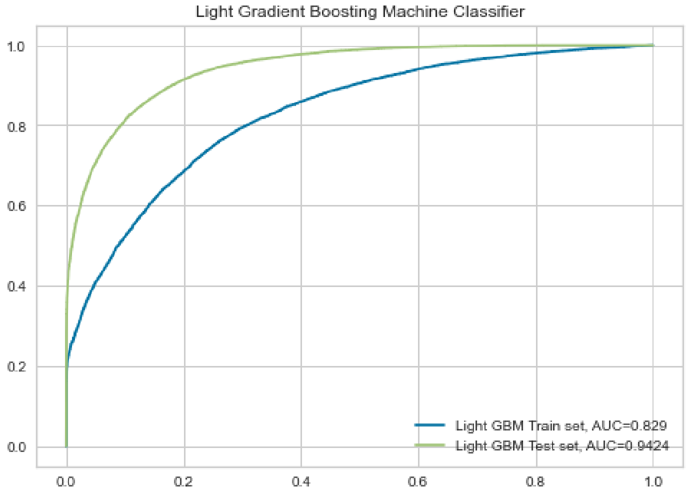
Choice: Light GBM



LIGHT GBM CLASSIFICATION (PREDICTIONS)

```python
from sklearn.model_selection import GridSearchCV

param_grid_class = {
    'boosting_type': ['gbdt'],
    'colsample_bytree': [1.0],
    'importance_type': ['split'],
    'learning_rate': [0.9],
    'max_depth': [-1],
    'min_child_samples': [20],
    'min_child_weight': [0.001],
    'min_split_gain': [0.0],
    'n_estimators': [220],
    'n_jobs': [-1],
    'num_leaves': [31],
    'random_state': [5266],
    'reg_alpha': [0.0],
    'reg_lambda': [0.0],
    'silent': ['warn'],
    'subsample': [1.0],
    'subsample_for_bin': [200000],
    'subsample_freq': [0]
}
```
✓ 0.4s

```python
optimal_params_class = GridSearchCV(
                        estimator = LGBM,
                        param_grid = param_grid_class,
                        scoring = 'f1',
                        verbose = 0,
                        cv = 3
)

optimal_params_class.fit(x_train,
                         y_train,
)

print(optimal_params_class.best_params_)
print(optimal_params_class.best_score_)
```
✓ 1.1s



AUC score (train set) = **0.829**

AUC score (test set) = **0.942**

Tuned with GridSearchCV:

- Learning rate = 0.9
- max_depth = -1
- n_estimators = 220
- random_state = 7801

Predictions (ranked based on Price of product)

	SKU_Number	Predicted_SoldFlag	MarketingType	ReleaseNumber	PriceReg	ReleaseYear	ItemCount_active
62037	3504396	1.0	S	8	317.00	2016	0
57196	3504561	1.0	S	5	274.99	2016	0
62022	3504528	1.0	S	7	274.99	2016	0
58159	3467973	1.0	S	2	250.00	2016	0
46327	2351093	1.0	S	5	160.00	2016	0
46051	3483275	1.0	S	2	149.95	2016	0
31645	1438250	1.0	S	8	116.75	2008	0
64494	858486	1.0	S	4	115.00	2015	0
48196	1440598	1.0	S	9	112.00	2011	0
62877	803858	1.0	S	2	89.95	2013	0

Total of 30 products needs to be restocked based on the Light GBM classification prediction.

CONCLUSION

Based on the Light GBM classification model's prediction:

I. There are **30 products** that have **high probability** of being **sold in the next 6 months.**

II. These **30 products are out of stock** in the **current inventory** and needs restocking.

III. By order of Price, the **top 5 products** cost at least **USD160.00** and were **released** quite **recently** (2016).

IV. Majority of the 30 products were **S-Marketing types** (28).

  I. **Earlier** EDA shows **S-Marketing and Soldcount** have **negative correlation.**

  II. Marketing and Sales might want to **dive deeper on the S-Marketed strategies** for these **28 products.**

LIMITATIONS AND TAKEAWAYS

LIMITATIONS

- Data dictionary regarding features

(such as Strength factor, Low User Price and Low Net Price)

- More characteristics about each unique product

(e.g. category, brand, made in <country>, etc.)

- Time series of sales across 6 months for more robust regressor model training

(how certain products or category of products would perform in the next quarter/3 months)

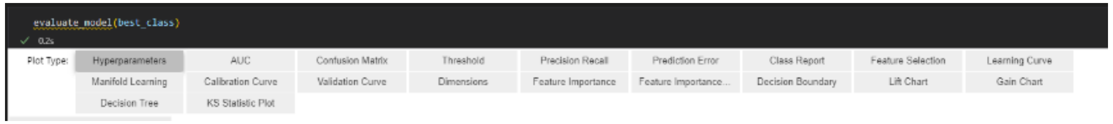
TAKEAWAYS

- Multiple steps and adjustments to train and refine a machine learning model (but not all need to be used)

- PyCaret is very useful

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
lightgbm	Light Gradient Boosting Machine	0.8086	0.6991	0.0513	0.5479	0.0937	0.0627	0.1217
gbc	Gradient Boosting Classifier	0.8078	0.7027	0.0431	0.5258	0.0796	0.0519	0.1067
ada	Ada Boost Classifier	0.8074	0.6949	0.0167	0.5463	0.0323	0.0210	0.0676
dummy	Dummy Classifier	0.8070	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
ridge	Ridge Classifier	0.8069	0.0000	0.0010	0.4367	0.0020	0.0011	0.0127
lr	Logistic Regression	0.8067	0.6795	0.0041	0.4256	0.0080	0.0045	0.0252
lda	Linear Discriminant Analysis	0.8066	0.6863	0.0082	0.4621	0.0162	0.0094	0.0396
xgboost	Extreme Gradient Boosting	0.8051	0.6948	0.0744	0.4713	0.1283	0.0799	0.1247
qda	Quadratic Discriminant Analysis	0.7957	0.6796	0.0789	0.3658	0.1296	0.0656	0.0910
nb	Naive Bayes	0.7904	0.6774	0.0949	0.3442	0.1487	0.0714	0.0909
knn	K Neighbors Classifier	0.7791	0.5971	0.1313	0.3235	0.1866	0.0845	0.0962
rf	Random Forest Classifier	0.7570	0.6252	0.2206	0.3152	0.2594	0.1196	0.1223
et	Extra Trees Classifier	0.7437	0.6049	0.2413	0.2978	0.2664	0.1133	0.1142
svm	SVM - Linear Kernel	0.7404	0.0000	0.1329	0.2351	0.0889	0.0273	0.0433
dt	Decision Tree Classifier	0.7261	0.5501	0.2616	0.2774	0.2691	0.1009	0.1010

- More to learn about evaluating models on PyCaret



THANK YOU

[GitHub link]