**Walmart Project: Trip Type Classification**

**Sandhiya Rajendran**

**Abstract**

Walmart uses both art and science to continually make progress on their core mission of better understanding and serving their customers. One way Walmart is able to improve customers' shopping experiences is by segmenting their store visits into different trip types.

The project aims to help Walmart classify customer trip type using the dataset of the items they’ve purchased. We apply various classification models on the purchase history data and the customer trip type provided by Walmart to solve the problem. The main challenges are transforming the raw data to features that represent trip type well and learning a predictive model based on these features.

**Problem statement and goals**

Walmart improves customers' shopping experiences by segmenting their store visits into different trip types. Whether they're on a last minute run for new puppy supplies or leisurely making their way through a weekly grocery list, classifying trip types enables Walmart to create the best shopping experience for every customer. Currently, Walmart's trip types are created from a combination of existing customer insights and purchase history data. In this problem, we will focus on the purchase history data and classify customer trips using only a transactional data set of the items they've purchased. The goal is to help Walmart refine their segmentation process by improving the data behind trip type classification. Walmart has categorized the trips contained in this data into **38 distinct types** with **647054 training samples** and **653646 test samples.**

**Data Fields:**

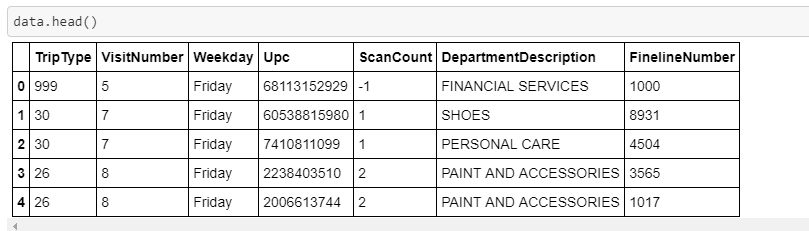
1. **TripType**-a categorical id representing the type of shopping trip the customer made. TripType\_999 is an "other" category.
2. **Visit Number**-an id corresponding to a single trip by a single customer
3. **Weekday**-the weekday of the trip
4. **UPC**-the UPC number of the product purchased
5. **Scan Count**-the number of the given item that was purchased. A negative value indicates a product return.
6. **Department Description**-a high-level description of the item's department
7. **Fineline Number**-a more refined category for each of the products, created by Walmart.

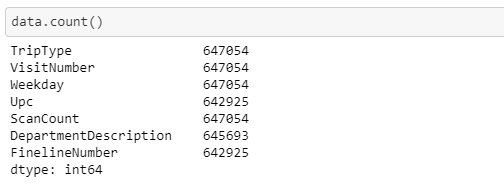
**Data Analysis**

**Exploring the Data**

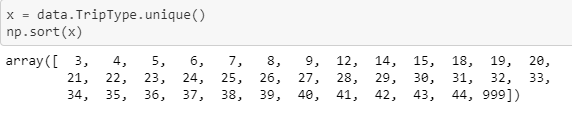
Snapshot of initial Data

Train

****

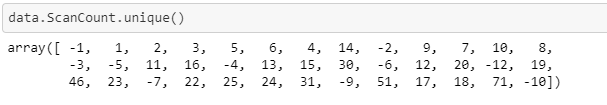


Trip Type:



There are 38 unique Trip Types where 999 represents the “other” category.

Scan Count:



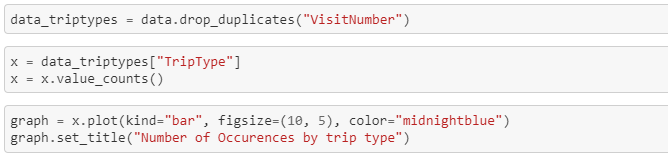


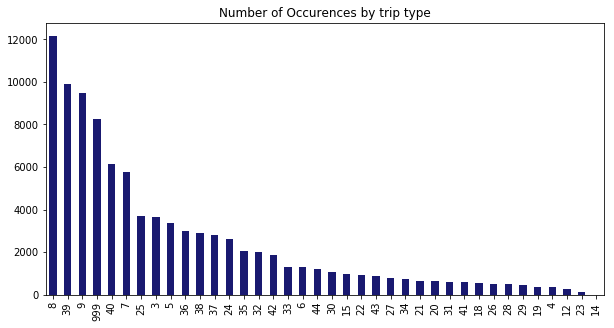
Here the negative values represent the return count. 71 looks like an outlier since it doesn’t fall in the range of other data.

Department description

There are 69 unique department descriptions.

Number of Occurrences by Trip Type.

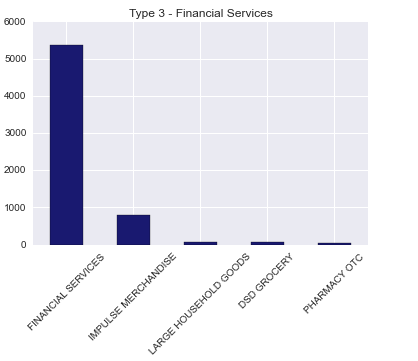




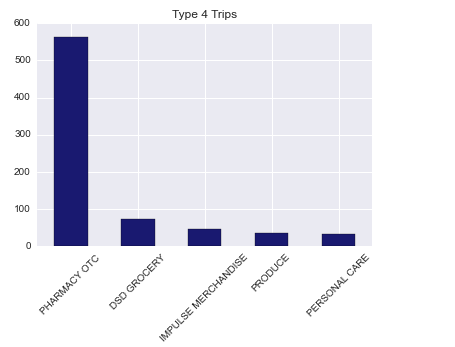
Interesting - occurrences of different trip types are not evenly distributed

Let’s try to visualize and categorize each trip type.

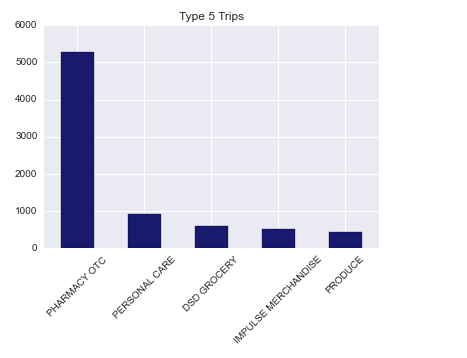
Trip 3 – Financial Services

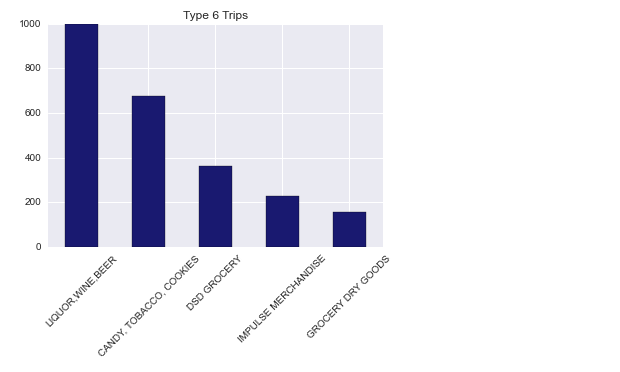


Trip 4 – Pharmacy OTC

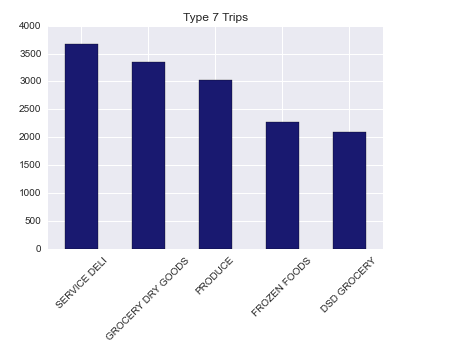


Trip 5– Pharmacy OTC

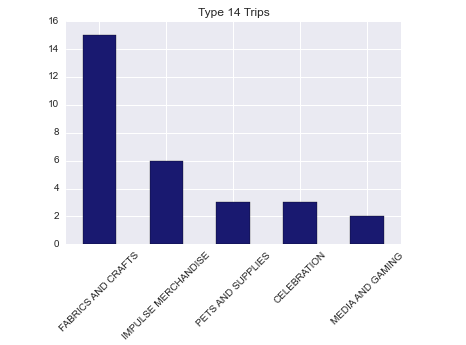


Trip 6– Alcohol + candy/tobacco/cookies + grocery 

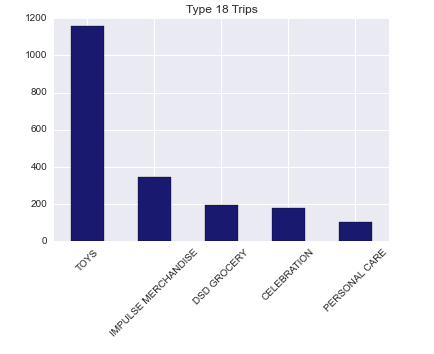
Trip 7 – Grocery trip



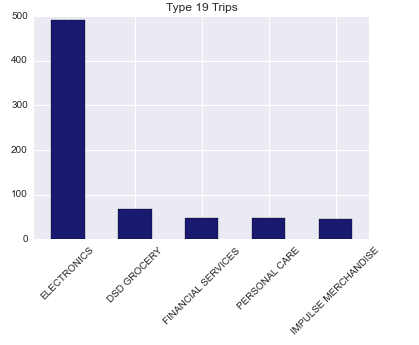
Trip 14 – Fabrics and Craft Trip



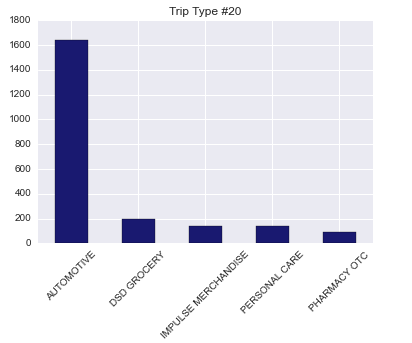
Trip 18 – Toys Trip



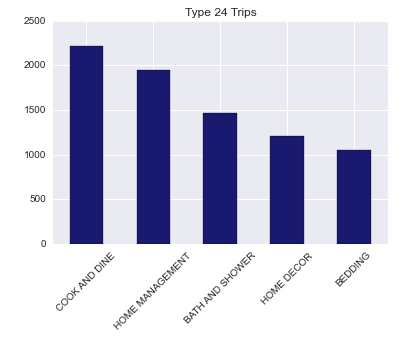
Trip 19 – Electronics Trip



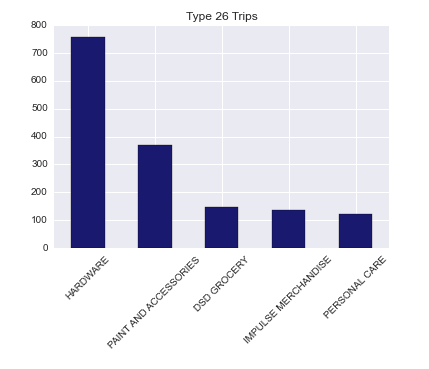
Trip 20- Automotive Trip



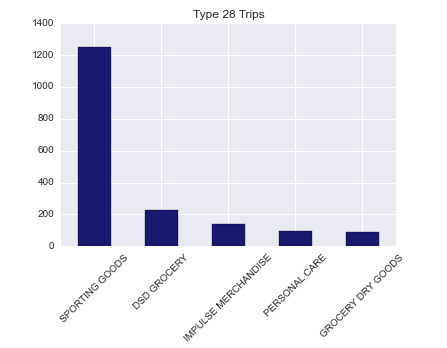
Trip 24 – Kitchen and Home Décor Trip



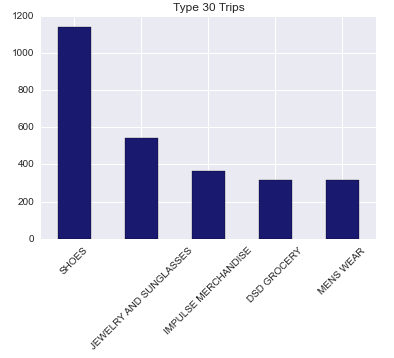
Trip 26 – Hardware and accessories Trip.



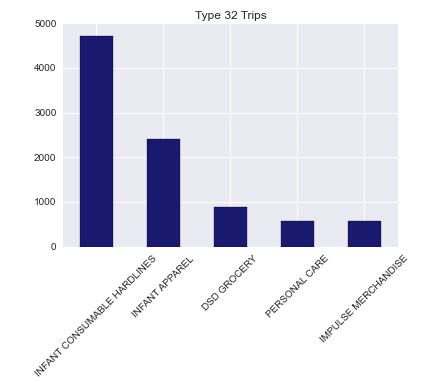
Trip 28 – Sports Trip



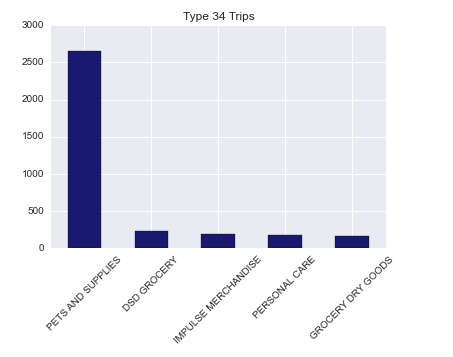
Trip 30 – Shoes Trip



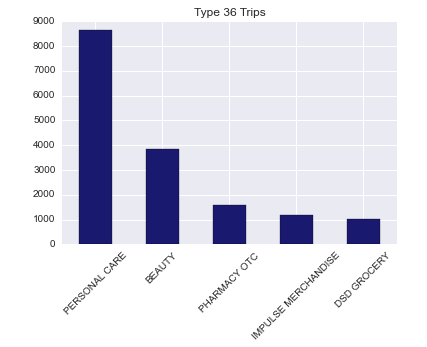
Trip 32 – Baby Products Trip



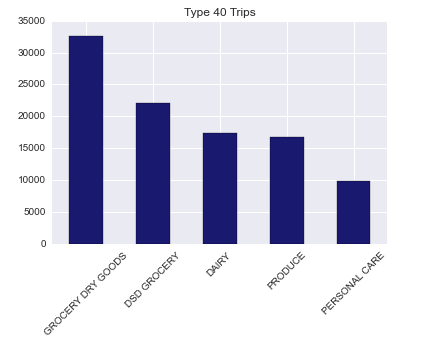
Trip 34 – Pets Trip



Trip 36: Personal Care Trip

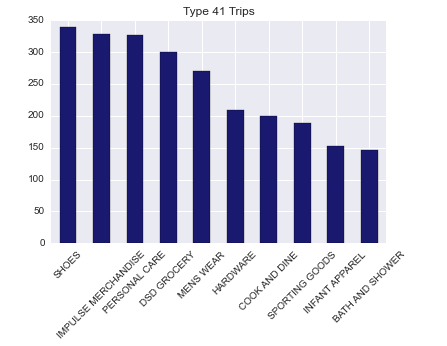


Trip 40 – Grocery Trip

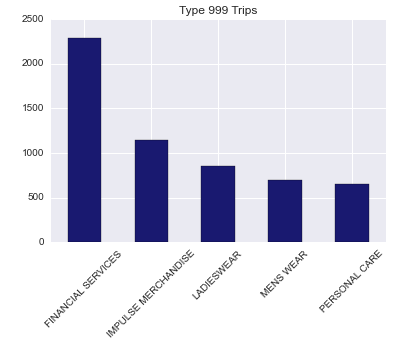


Trip 41 – Mix of many trips (Shoes + Personal care + grocery +kitchen + sports + Infant)

(Unusual trip)

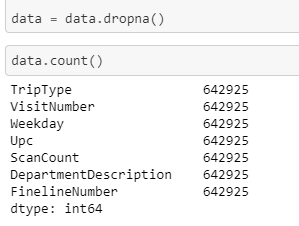


Trip 999 – Others (Mostly Financial)



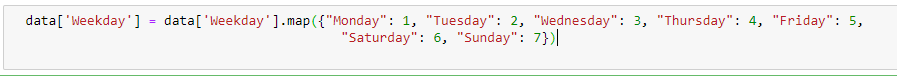
**Data Preprocessing**

1. Dropping rows with missing values

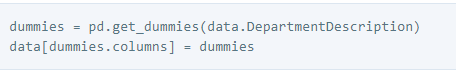


|  |  |  |  |
| --- | --- | --- | --- |
|  | Original Data Size | Missing data size | New data size |
| Training set | 647054 | 4129 | 642925 |
| Test set | 653646 | 3986 | 649660 |

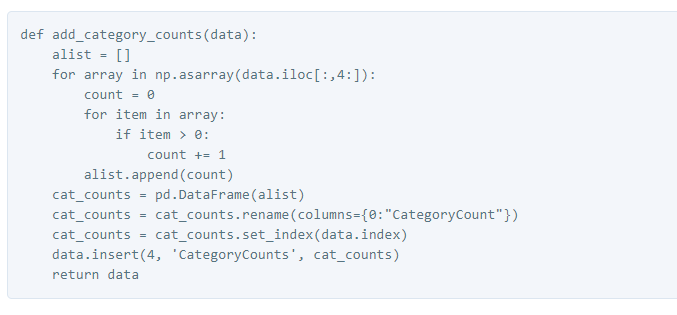
1. Numerate days of week



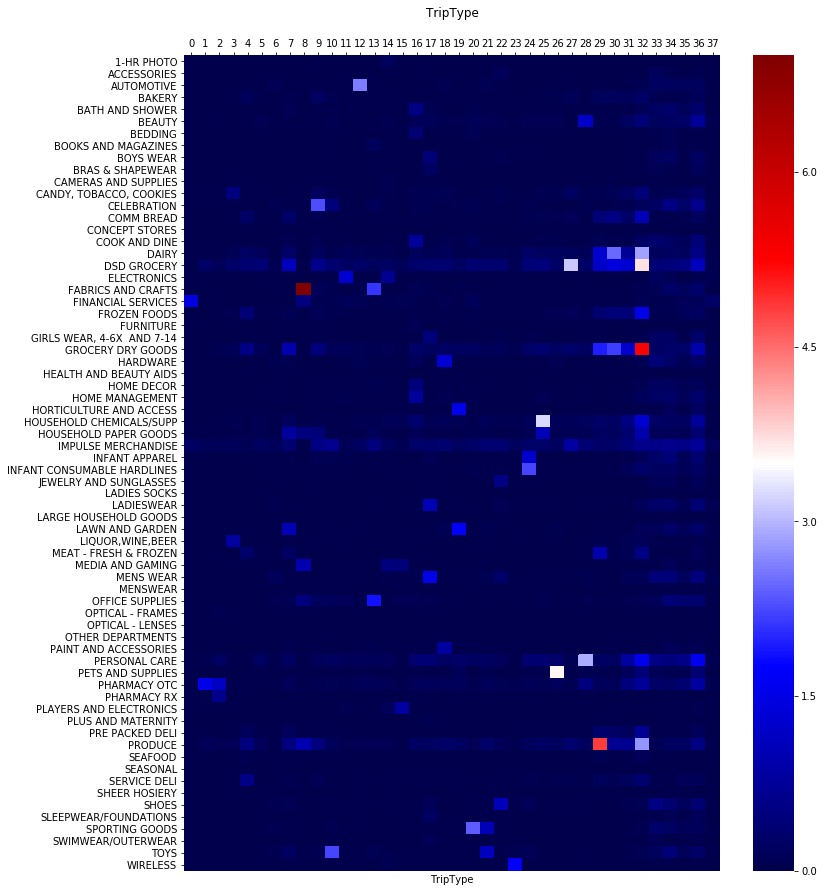
1. To describe the purchased item in a numerical way I created a (dummy) column for each Department description with the data being the amount purchased in that category per trip.



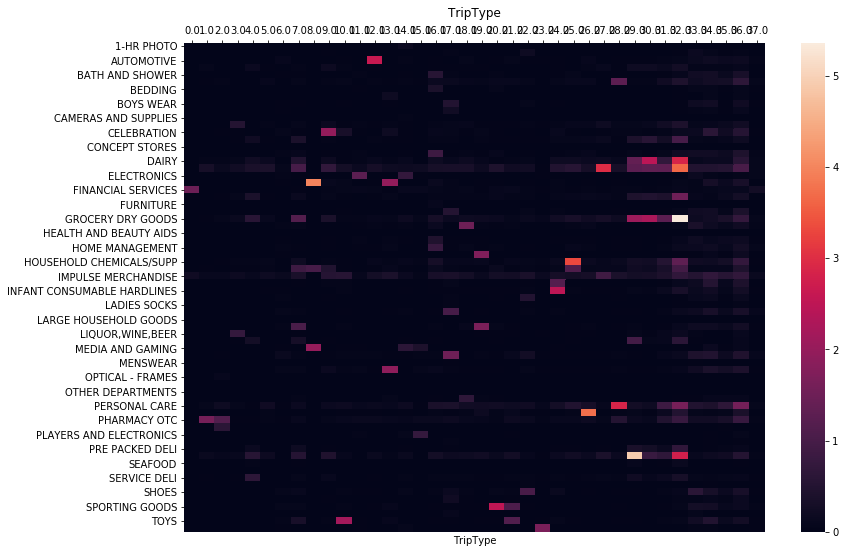
1. Category counts – total number of unique department purchases made on each trip. This signifies whether this was a multi- purpose or single-purpose trip.



Visualizing the departments and their trip types through a heatmap



Heatmap for limited number of Department description.



**Classification Models:**

**Model1: Random forest**

**Log loss Score:** 1.30534

**Accuracy Score:** 0.6740789129866737

**Precision Score:** [0.7994382 0.06666667 0.74974671 0.69960474 0.67223199 0.78502509

0.70432692 0.14285714 0. 0.54054054 0.43956044 0.24285714

0.67336683 0.657277 0.45089286 0.42 0.58123791 0.64722864

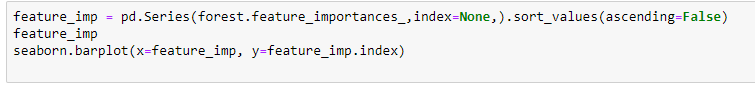
0.5234375 0.65480427 0.42038217 0.27777778 0.47923323 0.69230769

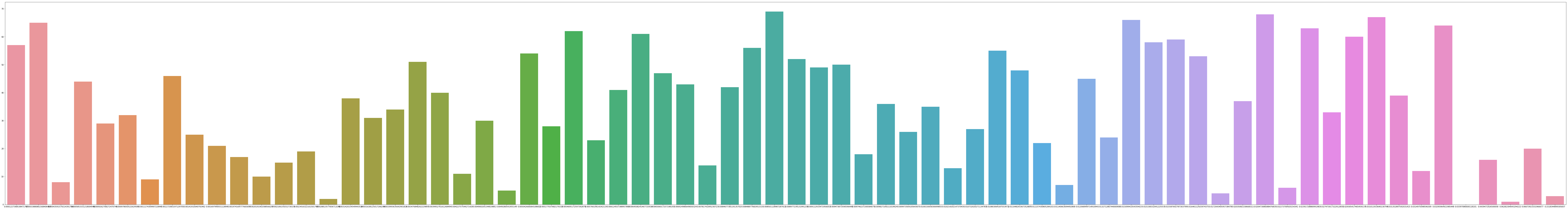
0.65207373 0.6617284 0.64224138 0.63256484 0.6209205 0.58344459

0.57964602 0.49376086 0.72781065 0.23076923 0.42295082 0.18181818

0.41666667 0.94742729]

Important feature graph





**Model2: Logistic Regression**

**Log loss Percentage:** 1.21611

**Accuracy Score:** 0.6540109746537758

**Precision Score:** [0.78481735 0. 0.70585626 0.59151786 0.64432342 0.67085549

0.63577725 0.25 0. 0.51229508 0.4893617 0.33333333

0.62910798 0.57079646 0.43005181 0. 0.58939096 0.6984326

0.49101796 0.68055556 0.46195652 0.2173913 0.43965517 0.67708333

0.66460396 0.6038961 0.59274194 0.60233918 0.60545309 0.59647495

0.51366743 0.53294875 0.80154308 0.23076923 0.48466258 0.

0.47787611 0.85679527]

**Model3: Naïve Bayes**

**Log loss Score:** 29.71509

**Accuracy Score:** 0.10185523909067154

**Precision Score:** [0.96666667 0.01143537 0.88188976 0.37948718 0.67368421 0.44827586

0.33333333 0.01100949 0. 0.06938776 0.07317073 0.03194888

0.3045977 0.0994152 0.2 0.01084641 0.46355685 0.63868613

0.11471698 0.40431267 0.06990521 0.02608696 0.24054463 0.09663866

0.57309942 0.46478873 0.41085271 0.15116279 0.45121951 0.30174927

0.15333333 0.42857143 0.75656566 0.13740458 0.45512821 0.0625

0.22673031 0.5

**Model4: Support Vector Machine.**

**Log loss Score:** 5.21626

**Accuracy Score**: 0.6493598118630781

**Precision Score**: [0.78850575 0. 0.68547368 0.58405172 0.64691047 0.65644172

0.63212095 0.23529412 0. 0.50378788 0.47715736 0.3372093

0.56768559 0.51515152 0.4 0. 0.56494096 0.65970655

0.48502994 0.61176471 0.45673077 0. 0.41917808 0.67156863

0.66628042 0.58736842 0.56382979 0.54964029 0.58519661 0.58572753

0.53958944 0.58255034 0.7743083 0. 0.46987952 0.

0.38461538 0.8246082 ]

**Recall Score**: [0.92204301 0. 0.88961749 0.52316602 0.66971279 0.85407286

0.75682057 0.03703704 0. 0.34725849 0.41777778 0.20138889

0.52845528 0.57627119 0.19390582 0. 0.61584158 0.78667564

0.39901478 0.63414634 0.47979798 0. 0.33849558 0.60352423

0.72327044 0.52641509 0.54266212 0.48110831 0.64158918 0.56783468

0.30846605 0.53846154 0.80286885 0. 0.20607662 0.

0.02283105 0.75098694]

**Model 5: KNN**

**Log loss Score:** 6.4758

**Accuracy Score:** 0.6290828325058793

**Precision Score:** [0.77207155 0.08247423 0.65638963 0.59844055 0.526413 0.70011907

0.66594098 0.10666667 0. 0.41916168 0.36238532 0.21333333

0.61214953 0.53012048 0.3083004 0.45833333 0.54252577 0.67354196

0.50793651 0.67266187 0.45384615 0.20408163 0.45306122 0.72395833

0.67792208 0.52767528 0.6064257 0.4648318 0.54624277 0.50951557

0.41795104 0.51413882 0.76487985 0.19230769 0.39726027 0.07936508

0.21428571 0.91663633]

**Recall score:** [0.89919355 0.05797101 0.80273224 0.59266409 0.62010444 0.84240688

0.72219307 0.07407407 0. 0.36553525 0.35111111 0.11111111

0.53252033 0.55932203 0.21606648 0.18333333 0.41683168 0.63728129

0.31527094 0.57012195 0.2979798 0.05617978 0.24557522 0.6123348

0.65660377 0.53962264 0.51535836 0.5743073 0.63905325 0.52920036

0.38642079 0.54590571 0.84795082 0.01992032 0.11492734 0.01453488

0.04794521 0.76465229]

**Model6: XGBoost**

**Log loss Score:** 0.98281

**Accuracy Score:** 0.6873007577737131

**Precision Score:** [0.78154825 0. 0.71589883 0.66166667 0.69557022 0.76694915

0.71515613 0.3125 0. 0.60493827 0.49152542 0.33628319

0.63855422 0.62650602 0.405 0.43333333 0.62581345 0.70499343

0.52873563 0.59248555 0.44444444 0.28571429 0.45901639 0.60895522

0.60938967 0.65052632 0.59581882 0.62907268 0.63805668 0.63378804

0.55201958 0.53922673 0.75540541 0.28125 0.45936396 0.

0.35294118 0.99482278]

**Recall score:** [0.9905914 0. 0.86612022 0.76640927 0.64229765 0.8520262

0.73898216 0.0462963 0. 0.38381201 0.51555556 0.26388889

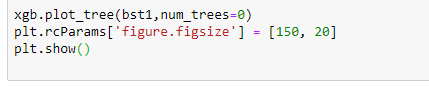
0.64634146 0.66101695 0.22437673 0.43333333 0.57128713 0.72207268

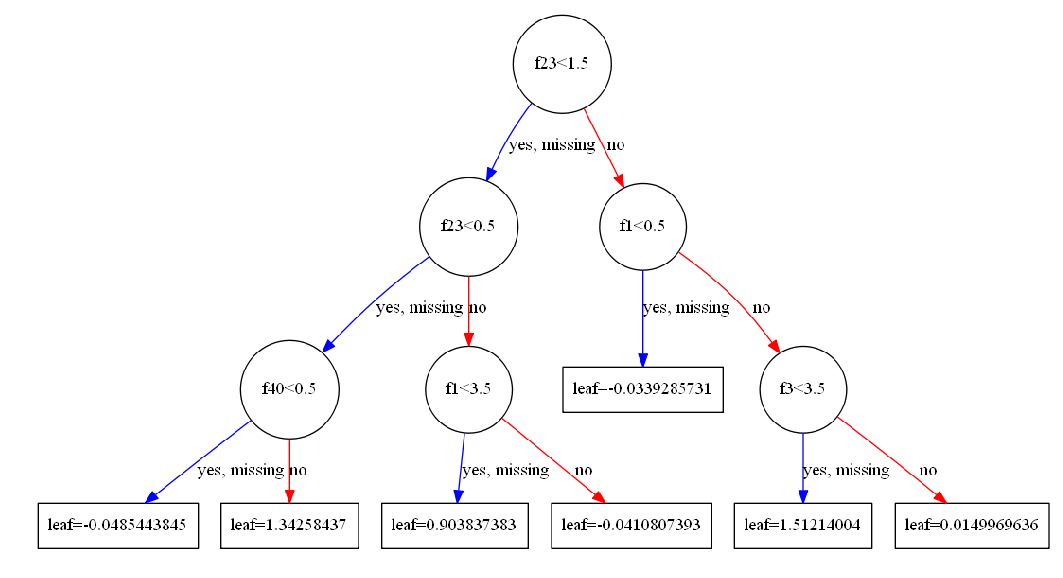
0.45320197 0.625 0.46464646 0.01123596 0.37168142 0.89867841

0.8163522 0.58301887 0.58361775 0.63224181 0.66610313 0.54267745

0.37803856 0.71290323 0.91639344 0.03585657 0.17173052 0.

0.06849315 0.7585788 ]





My rank for the model based on accuracy.

|  |  |  |
| --- | --- | --- |
| Model | Accuracy | Rank |
| Random Forest | 0.6740789129866737 | 2 |
| Naïve Bayes | 0.10185523909067154 | 6 |
| Support Vector Machine | 0.6493598118630781 | 4 |
| KNN Classifier | 0.6290828325058793 | 5 |
| XGBoost | 0.7273007577737131 | 1 |
| Logistic Regression | 0.6540109746537758 | 3 |

XGBoost has the highest accuracy of all models so I choose this model to test the test dataset.

**XGBoost Model on Test data**

**Log loss Score:** 0.91733

**Accuracy Score:** 0.7081525999477397

**Precision Score:** [0.78302387 0. 0.73865784 0.70856102 0.70577105 0.76474833

0.7281999 0.84615385 0. 0.66666667 0.51652893 0.51351351

0.72522523 0.69379845 0.50381679 0.53731343 0.68451025 0.72809864

0.59782609 0.64655172 0.50490196 0.6875 0.51089588 0.63496933

0.64509804 0.68987342 0.67491166 0.66391185 0.67087542 0.66173362

0.57945042 0.55338046 0.77260367 0.7027027 0.62580645 0.5

0.65277778 0.9979992 ]

**Recall score:** [0.99193548 0. 0.85409836 0.75096525 0.64926023 0.86758084

0.74921301 0.10185185 0. 0.41775457 0.55555556 0.26388889

0.65447154 0.75847458 0.36565097 0.6 0.5950495 0.75504711

0.54187192 0.68597561 0.52020202 0.06179775 0.46681416 0.91189427

0.82767296 0.61698113 0.65187713 0.6070529 0.6737109 0.56244385

0.40653814 0.75756824 0.93155738 0.10358566 0.25627477 0.00290698

0.10730594 0.75736411]

Accuracy comparison on test and train data.

|  |  |
| --- | --- |
|  | XGBoost Accuracy |
| Training data | 0.7273007577737131 |
| Test data | 0.7081525999477397 |

**Conclusion**

There seems to be a good correlation between the training data and test data and the error is minimal between these two. Despite the huge dataset with **647054 training samples** and **653646 test samples**, the maximum accuracy was achieved i.e. about 72 % with XGBoost classification model. Through trial and error with several models we have arrived at the best model and it has provided a good learning curve throughout this project.