LKH'S CAPSTONE: ADOPTION LIKELIHOOD

Dataset Size: 10290 rows & 24 columns

WHY?

4

Shop

Funds will support our animal welfare services that cost $3 million a year to run.

Learn more ›

today

Singapore World Big Read Gen Y Speaks Adulting 101 Commentary Voices Videos Brand Spotlight 8 DAYS

Most S'pore animal shelters full as groups cancel pet adoption drives amid Covid-19 outbreak

By MANDY LEE

Published MA

Updated MA

94

THE STRAITS TIMES LIFE

More people in Singapore interested in adopting or fostering pets during Covid-19 pandemic

Undergraduate Sarah Chua and her family adopted Tau Pok during the circuit breaker period. PHOTO: COURTESY OF SARAH CHUA

FORMAT:

1) EDA (Knowing your data) & Data preprocessing

- Non-graphical approach
- Univariate analysis eg. Null data
- Multivariate analysis eg. Pairplot
- Interesting facts

Includes feature engineering and encoding during the analysis

2) Machine learning

- Handling imbalanced data (SMOTETomek)
- Feature analysis before plugging into model
- Auto ML to preview which models work well
- Data Scale: non scaled VS standardized VS normalized
- Models & Metrics
- Hyperparameter tuning & Metrics

Also includes Visualizations

EDA:
KNOWING YOUR DATA

Non-Graphical Analysis

1) Highly Categorical -> Objects more than half of all data
2) Highly Cardinal data -> up to 7k unique values almost 70% unique.

```
[11]: adoption.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10290 entries, 0 to 10289
Data columns (total 24 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   id                10290 non-null  int64
 1   intakedate        10290 non-null  object
 2   intakereason      10288 non-null  object
 3   istransfer        10290 non-null  int64
 4   sheltercode       10290 non-null  object
 5   identichipnumber  8324 non-null   object
 6   animalname        10290 non-null  object
 7   breedname         10245 non-null  object
 8   basecolour        10290 non-null  object
 9   speciesname       10290 non-null  object
 10  animalage         10290 non-null  object
 11  sexname           10290 non-null  object
 12  location          10290 non-null  object
 13  movementdate      10290 non-null  object
 14  movementtype      10290 non-null  object
 15  istrial           10289 non-null  float64
 16  returndate        3256 non-null   object
 17  returnedreason    10290 non-null  object
 18  deceaseddate      326 non-null    object
 19  deceasedreason    10290 non-null  object
 20  diedoffshelter    10290 non-null  int64
 21  puttosleep        10290 non-null  int64
 22  isdoa             10290 non-null  int64
 23  label             10290 non-null  int64
dtypes: float64(1), int64(6), object(17)
memory usage: 1.9+ MB
```

no. of Unique values of the data

```
adoption.nunique()

id                  7288
intakedate          5306
intakereason          25
istransfer             2
sheltercode         7288
identichipnumber    5460
animalname          4336
breedname            799
basecolour            78
speciesname           27
animalage            273
sexname                3
location              39
movementdate         872
movementtype           7
istrial                1
returndate           756
returnedreason        24
deceaseddate         211
deceasedreason        13
diedoffshelter         2
puttosleep             2
isdoa                  2
label                  3
dtype: int64
```

EDA: KNOWING YOUR DATA

Non-Graphical Analysis

Number of variables	24
Number of observations	10290
Missing cells	19012
Missing cells (%)	7.7%
Duplicate rows	0
Duplicate rows (%)	0.0%

identichipnumber has 1966 (19.1%) missing values

returndate has 7034 (68.4%) missing values

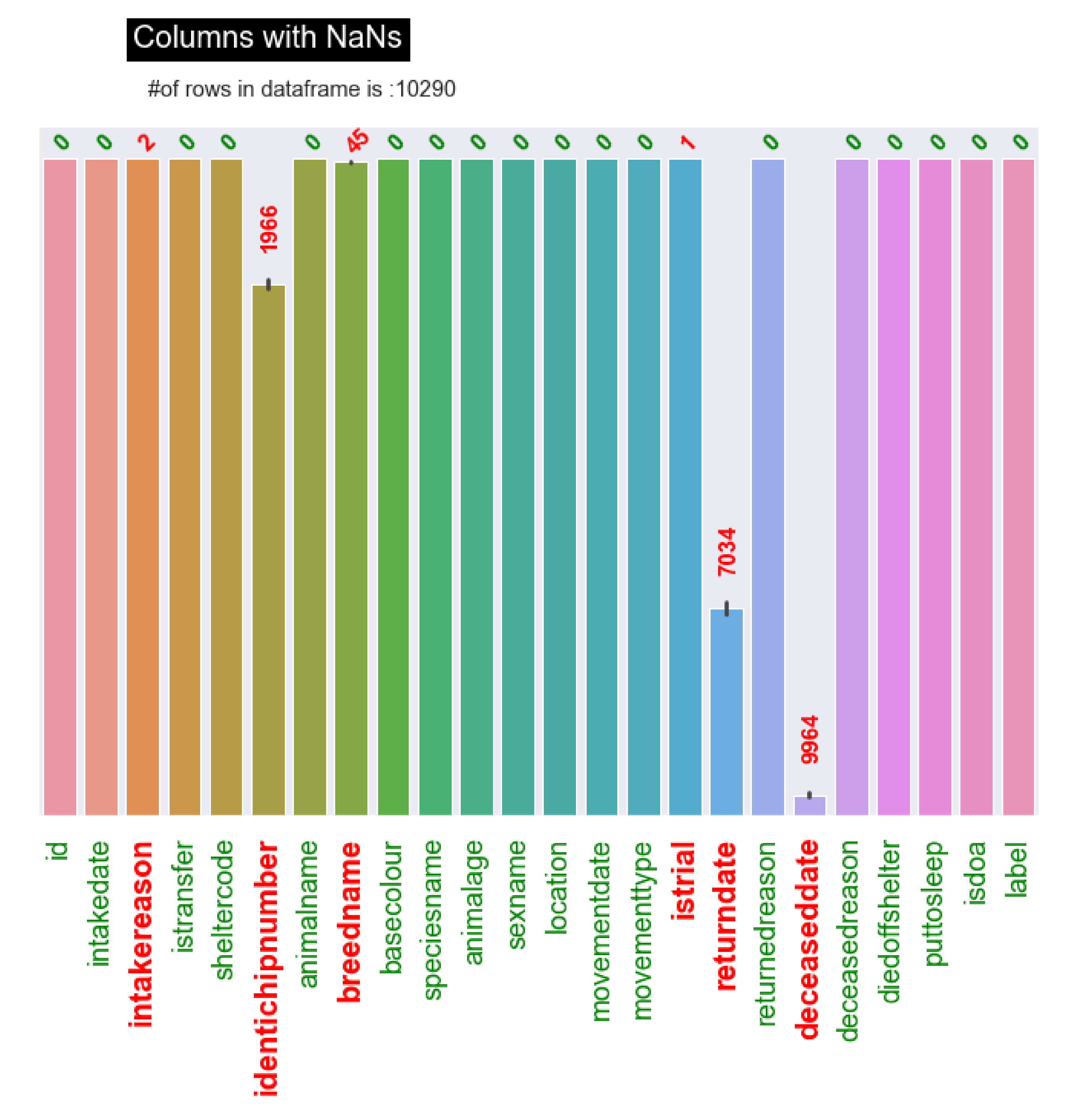
deceaseddate has 9964 (96.8%) missing values

Numeric	1
Categorical	23

istrial has constant value "0.0"	Constant
intakedate has a high cardinality: 5306 distinct values	High cardinality
sheltercode has a high cardinality: 7288 distinct values	High cardinality
identichipnumber has a high cardinality: 5460 distinct values	High cardinality
animalname has a high cardinality: 4336 distinct values	High cardinality
breedname has a high cardinality: 799 distinct values	High cardinality
basecolour has a high cardinality: 78 distinct values	High cardinality
animalage has a high cardinality: 273 distinct values	High cardinality
movementdate has a high cardinality: 872 distinct values	High cardinality
returndate has a high cardinality: 756 distinct values	High cardinality
deceaseddate has a high cardinality: 211 distinct values	High cardinality

1) Highly Categorical -> Objects more than half of all data
2) Highly Cardinal data -> up to 7k unique values almost 70% unique.

Key takeaway:
Require alot of feature engineering to reduce high cardinality



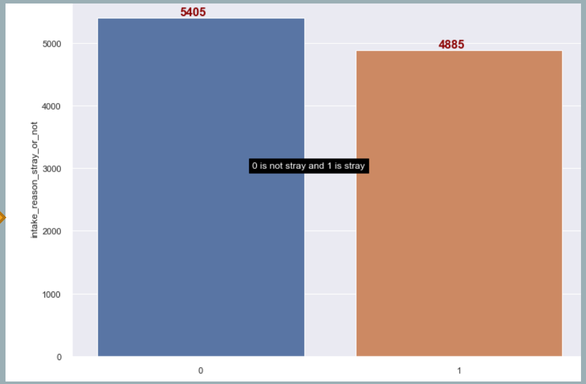
EDA:
KNOWING YOUR DATA

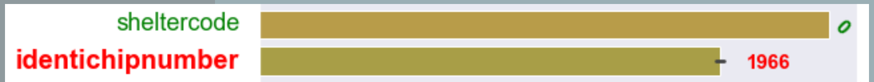
Univariate Analysis

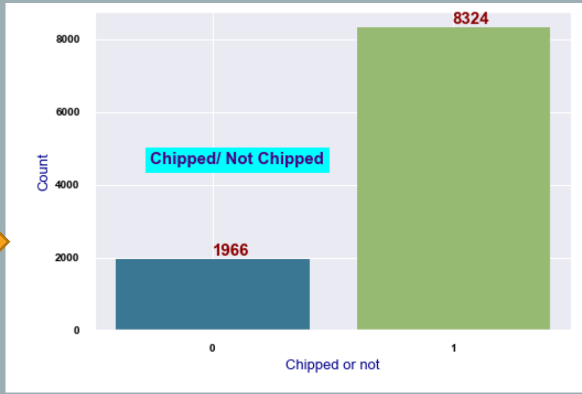
Missing values ->
drop or feature engineer
-> There is no numerical value to impute a mean etc.

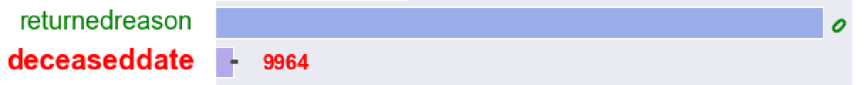
Important thing to note is that returndate and deceased date is missing but their respective reasons are not!

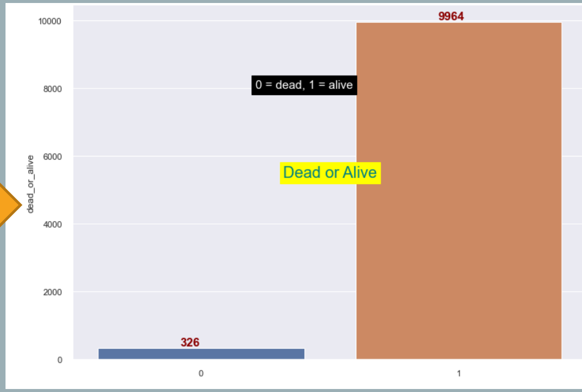
EDA: KNOWING YOUR DATA







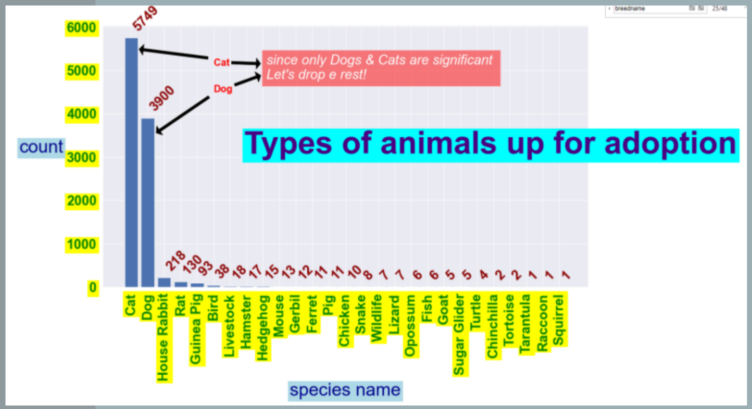


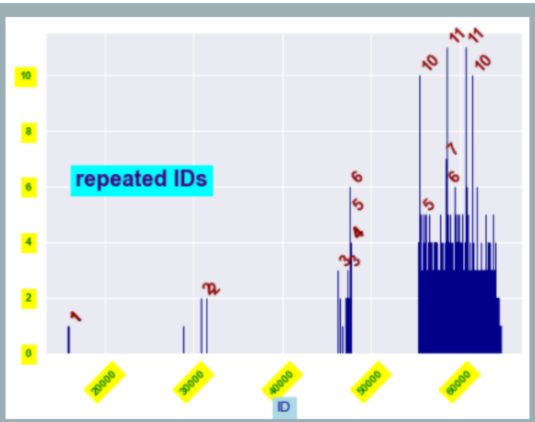


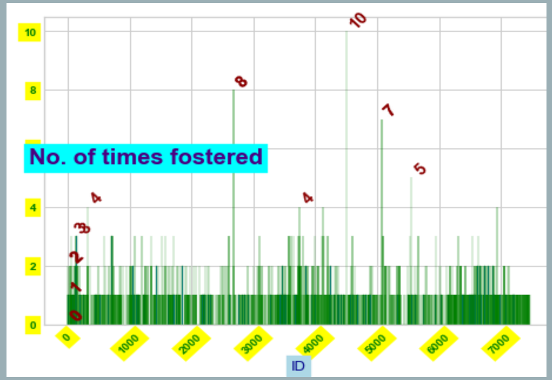
Univariate Analysis:
Dealing with Nulls

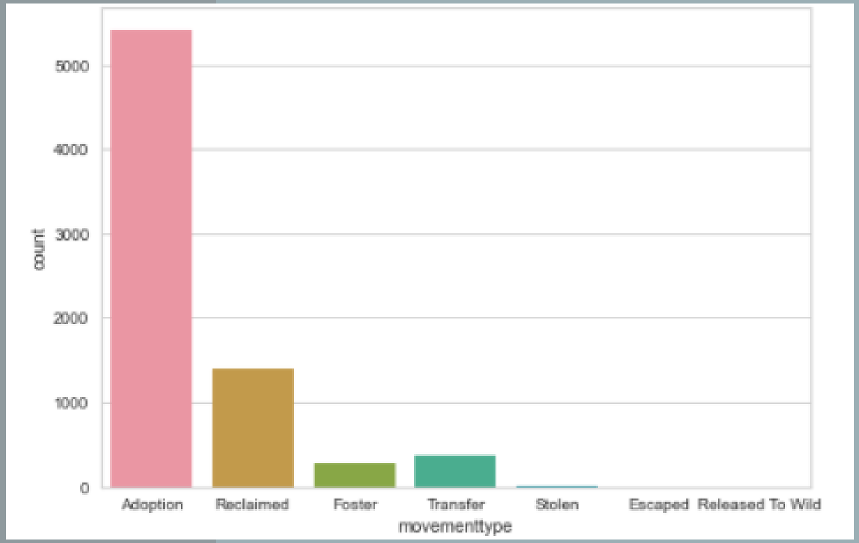
Feat. Engineering:
1) Aggregation
2) a little bit of Creativity eg. Missing means not dead etc.

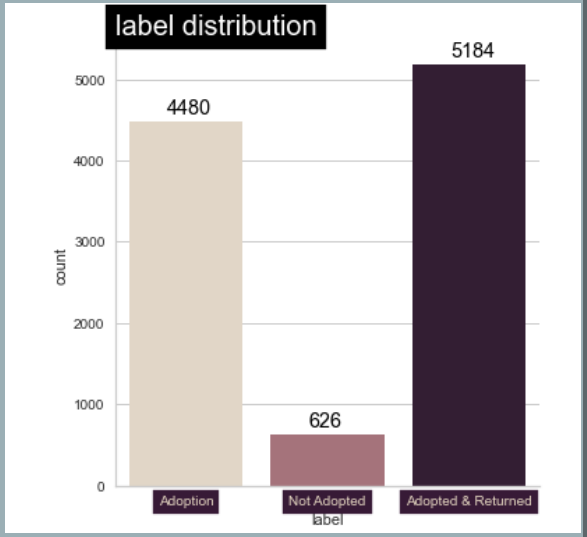
Key takeaway:
- Strays are the main reason why animals are in the shelter
- Both having an identity chip & being alive is important to determine if animal is to be adopted.











EDA: KNOWING YOUR DATA

Univariate Analysis:

What some of the other columns look like

Other important features:

1) mostly cats & dogs

2) There are repeated IDs -> Feature engineer into no. Of times fostered

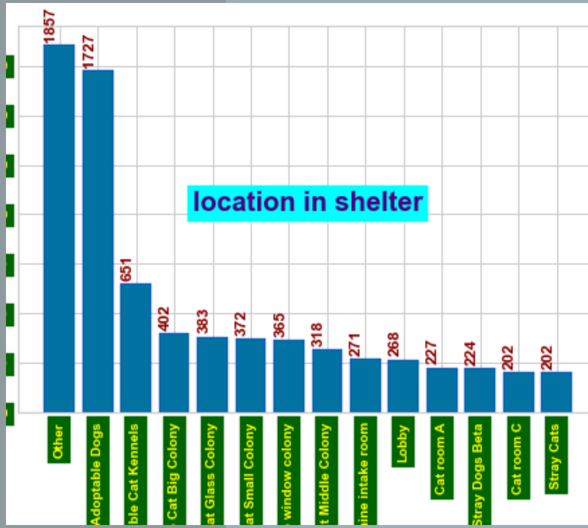
3) in general converting most of my columns into binaries via aggregation

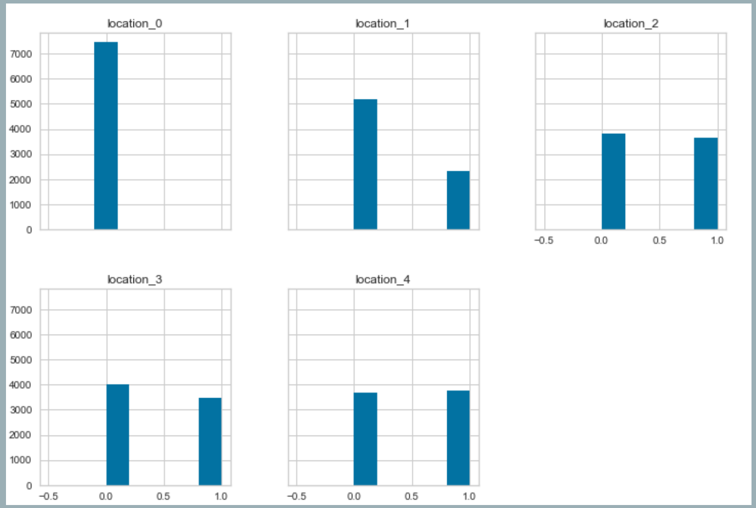
EDA:
KNOWING YOUR DATA

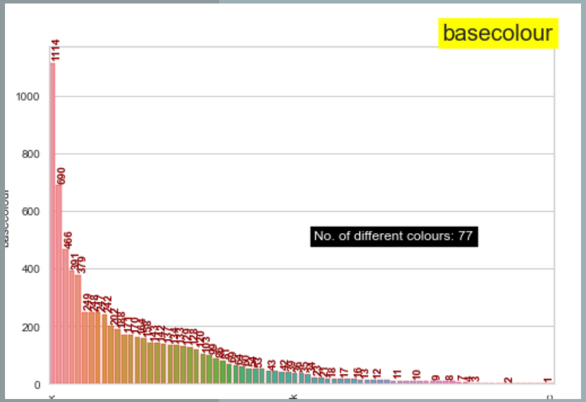
Univariate Analysis:
What some of the other columns look like

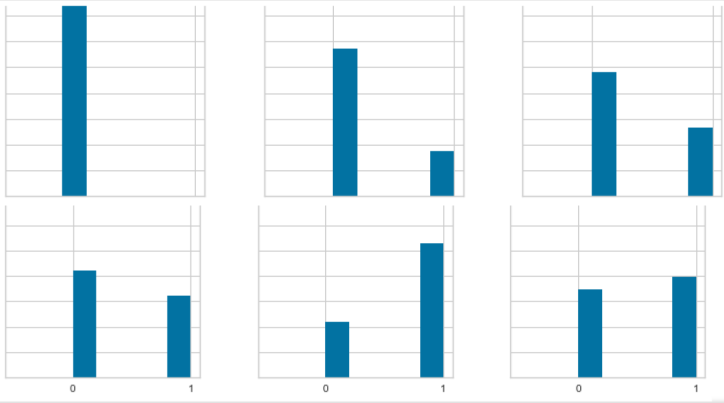
Binary encoding to avoid curse of dimensionality

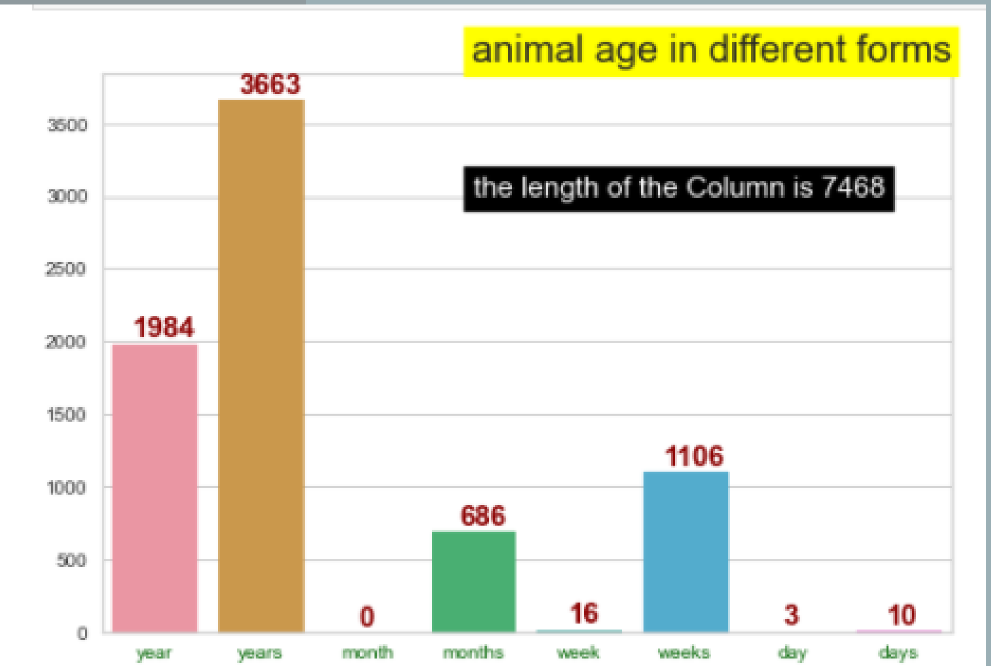
Not as many as one hot
But still able to reduce no. Of columns

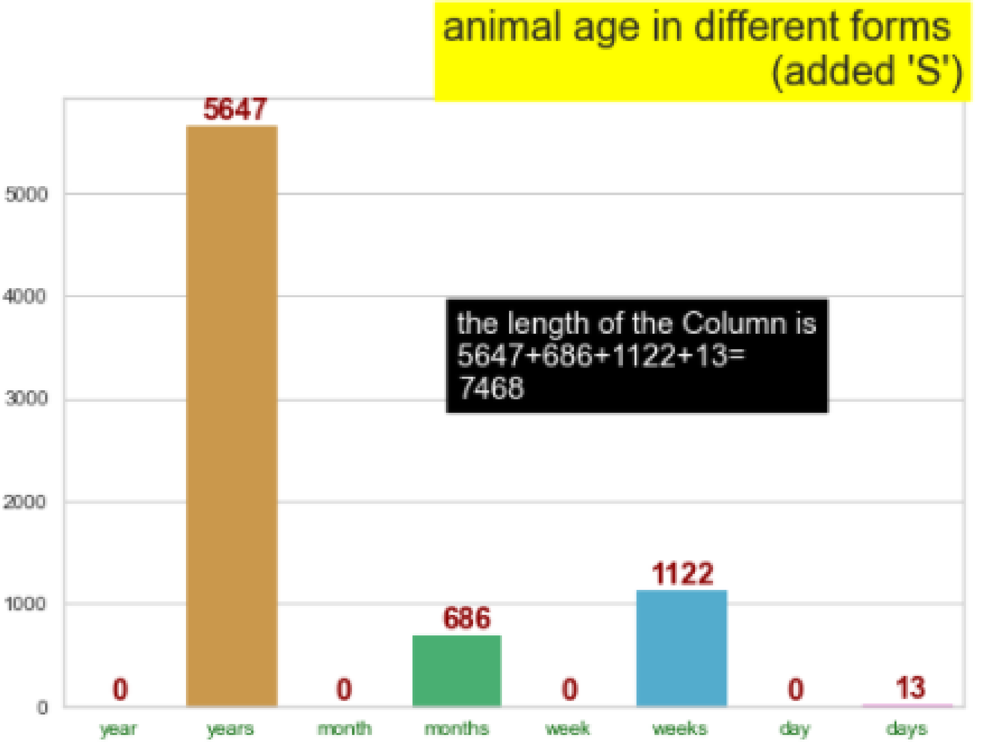


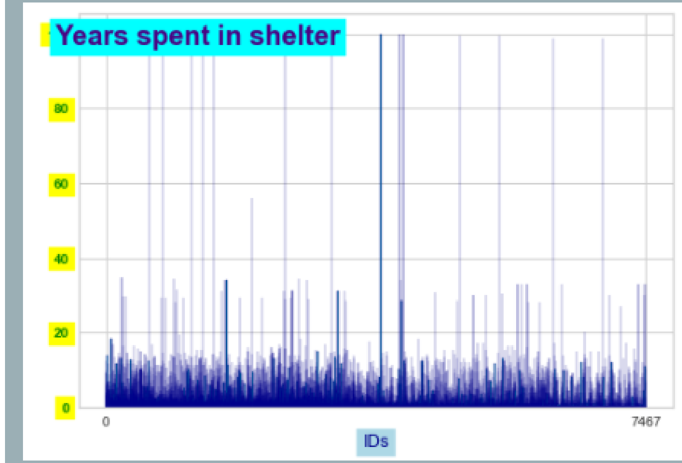


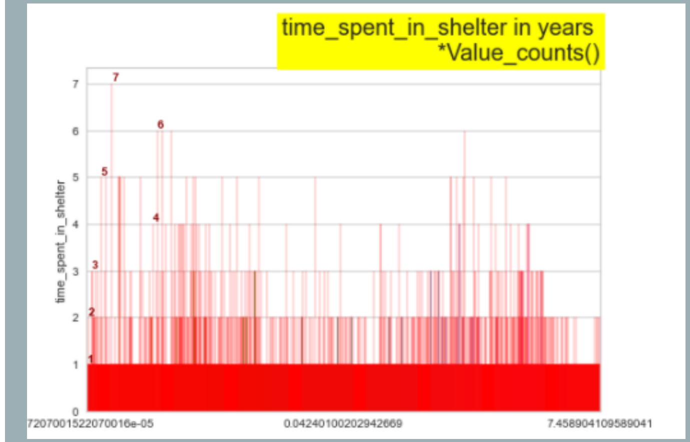










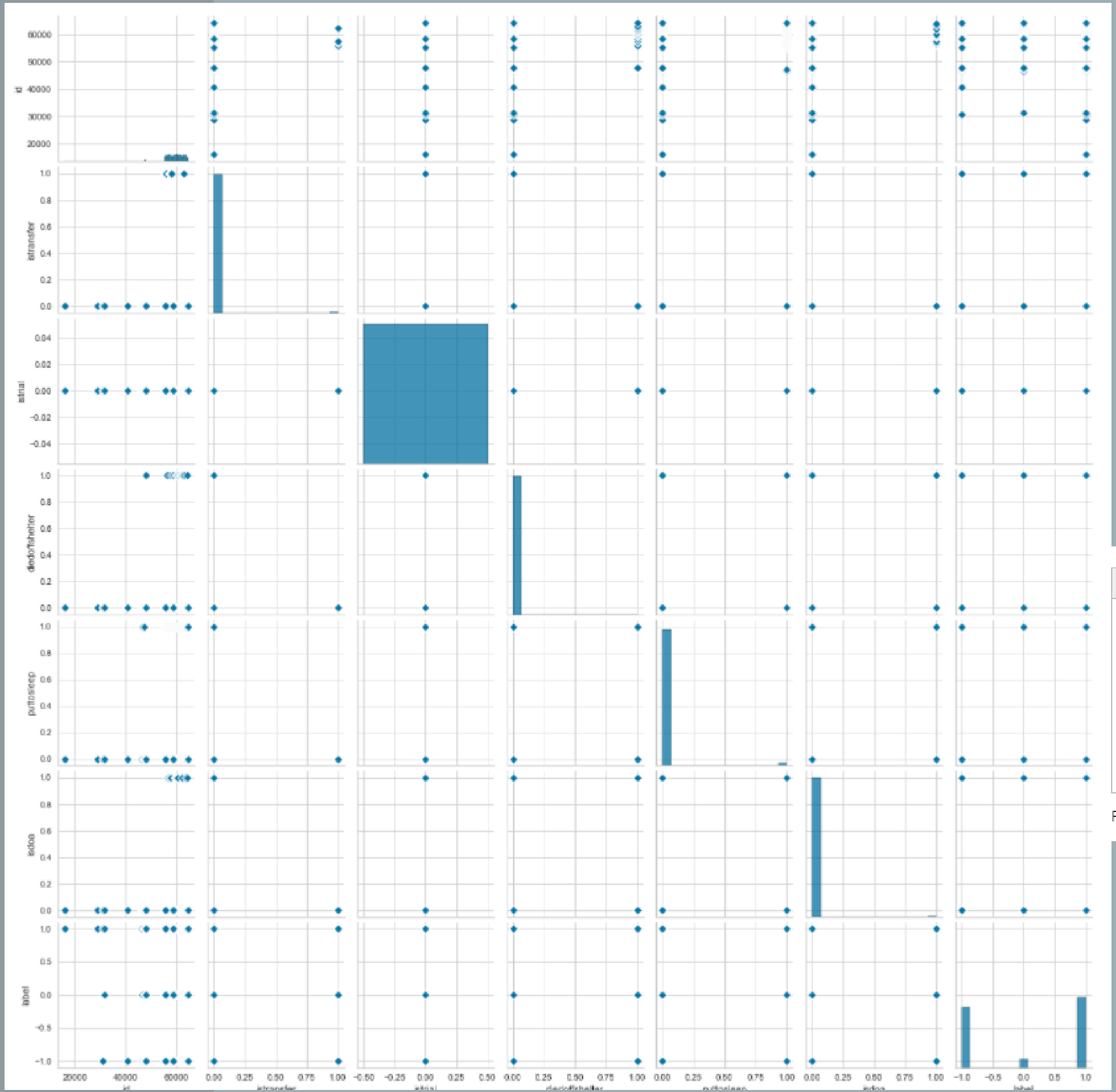


EDA:
KNOWING YOUR DATA

Univariate Analysis:
More challenging columns

Most challenging column was the date time column.

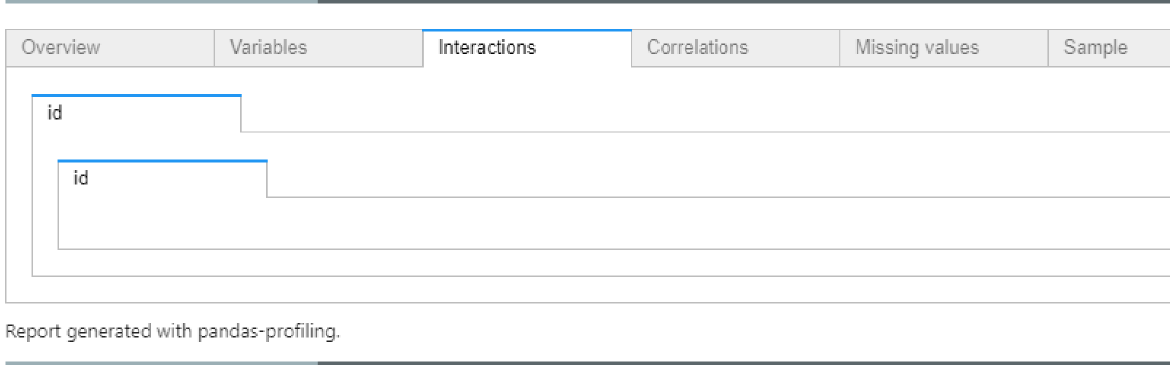
I simplified the column into years



EDA:
KNOWING YOUR DATA

Multivariate Analysis

Both the pairplot and the auto-EDA library both agree that there is nothing to plot



Key takeaway:
No numerical/ continuous data to plot with so lets create some

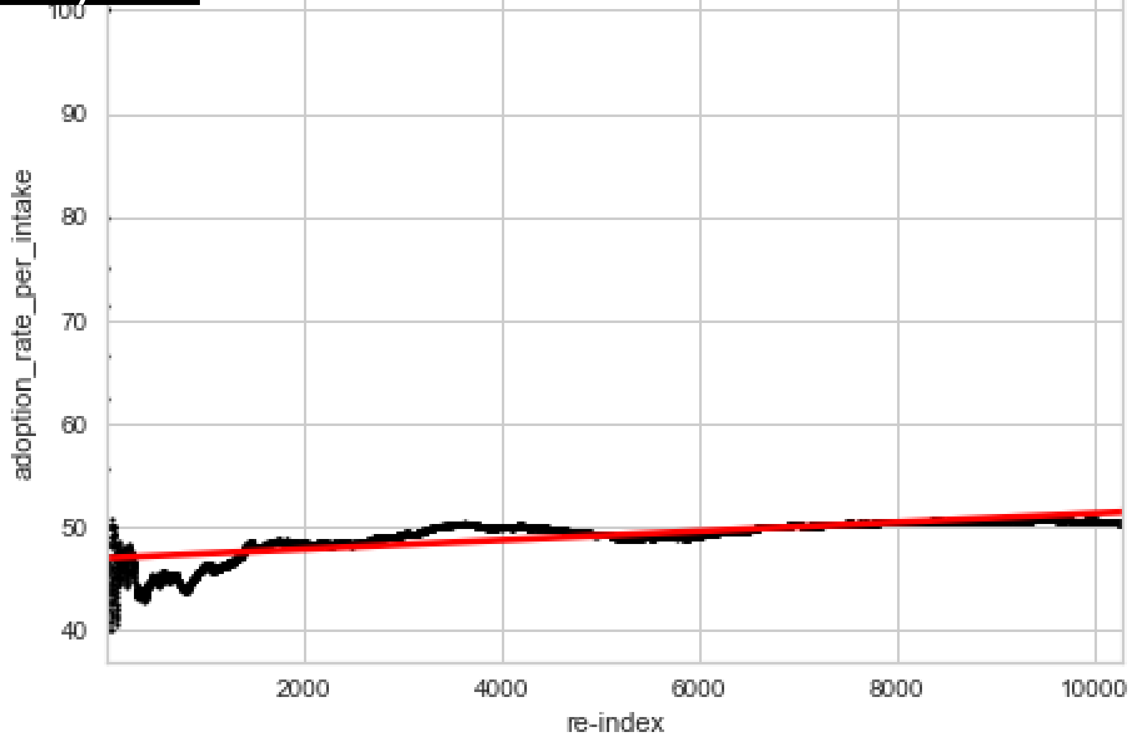
Both the graphs we can tell that adoption is not time sensitive

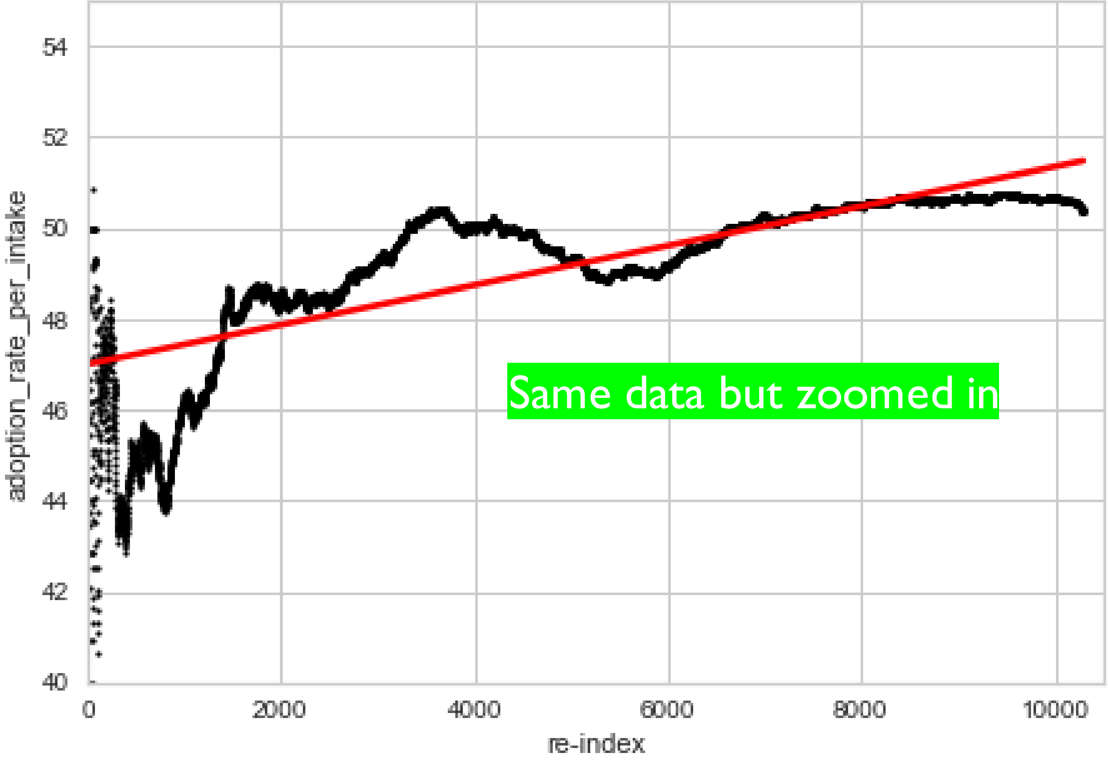
And that most of the data comes from after 2017

EDA: KNOWING YOUR DATA

Univariate Analysis

Adoption rate over each instance of adoption ordered by time

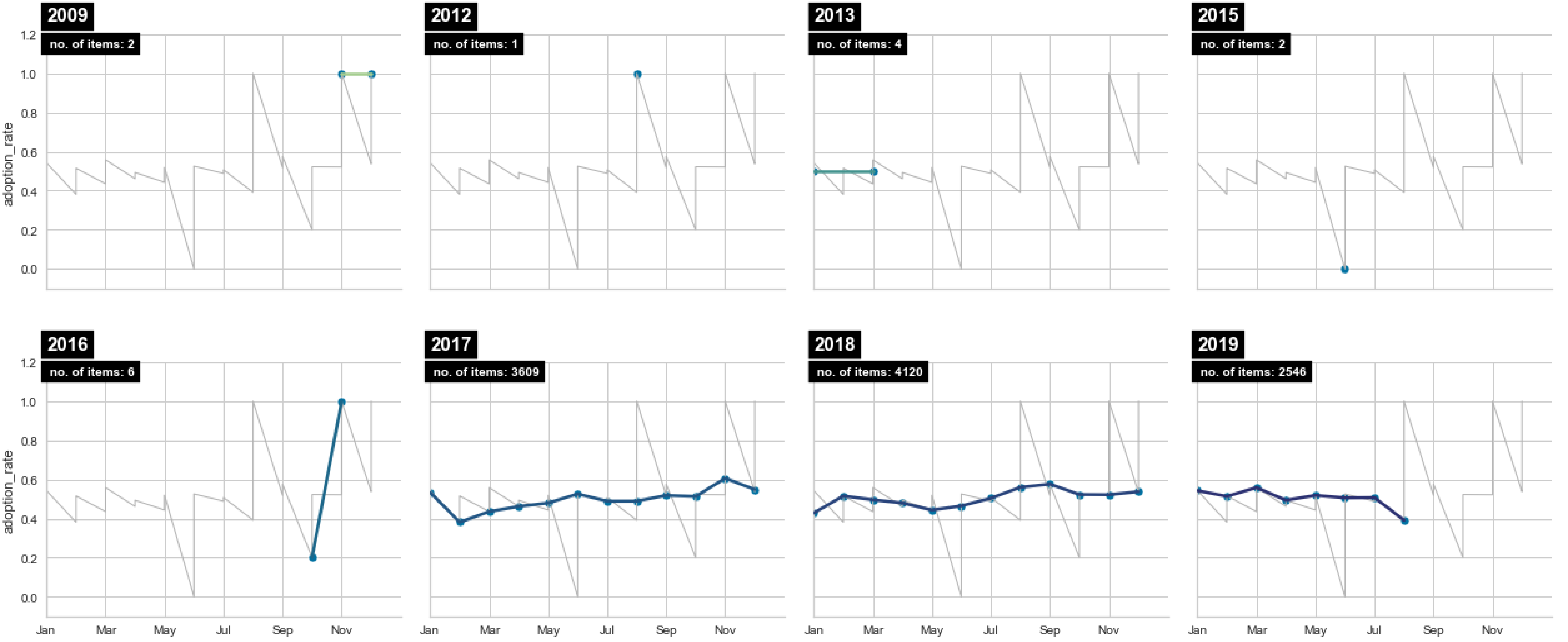




Zoomed in graph

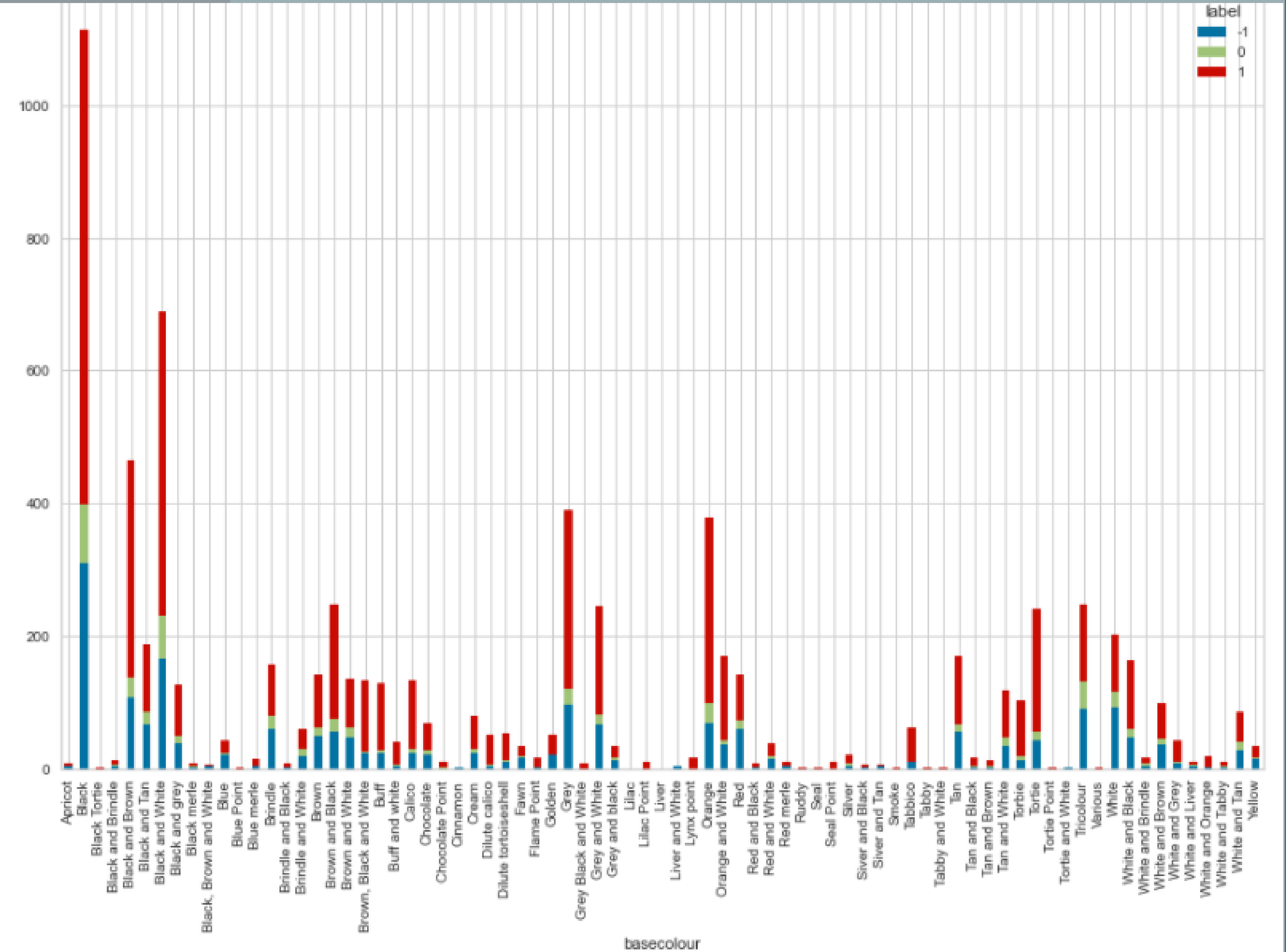
EDA: KNOWING YOUR DATA

Adoption rate Over the years per month

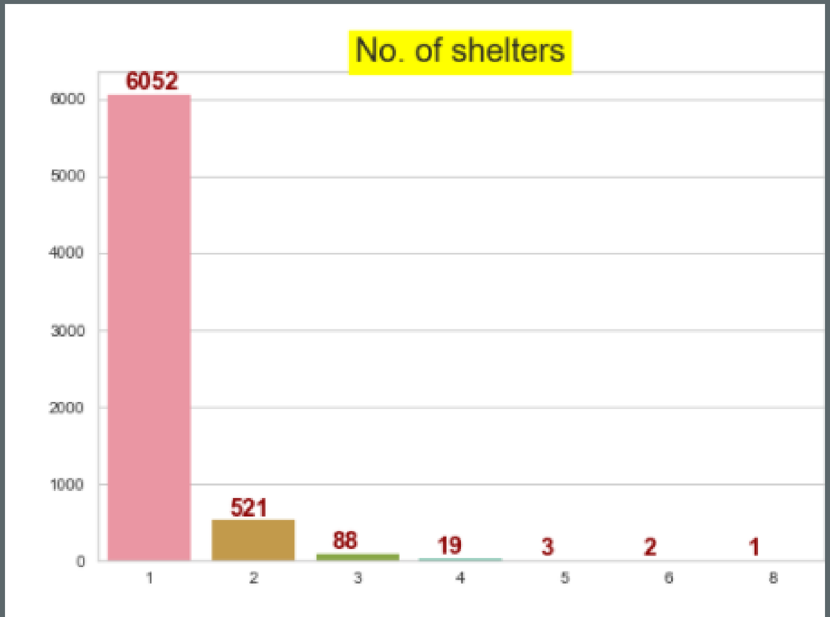


EDA:
KNOWING YOUR DATA

Other interesting facts

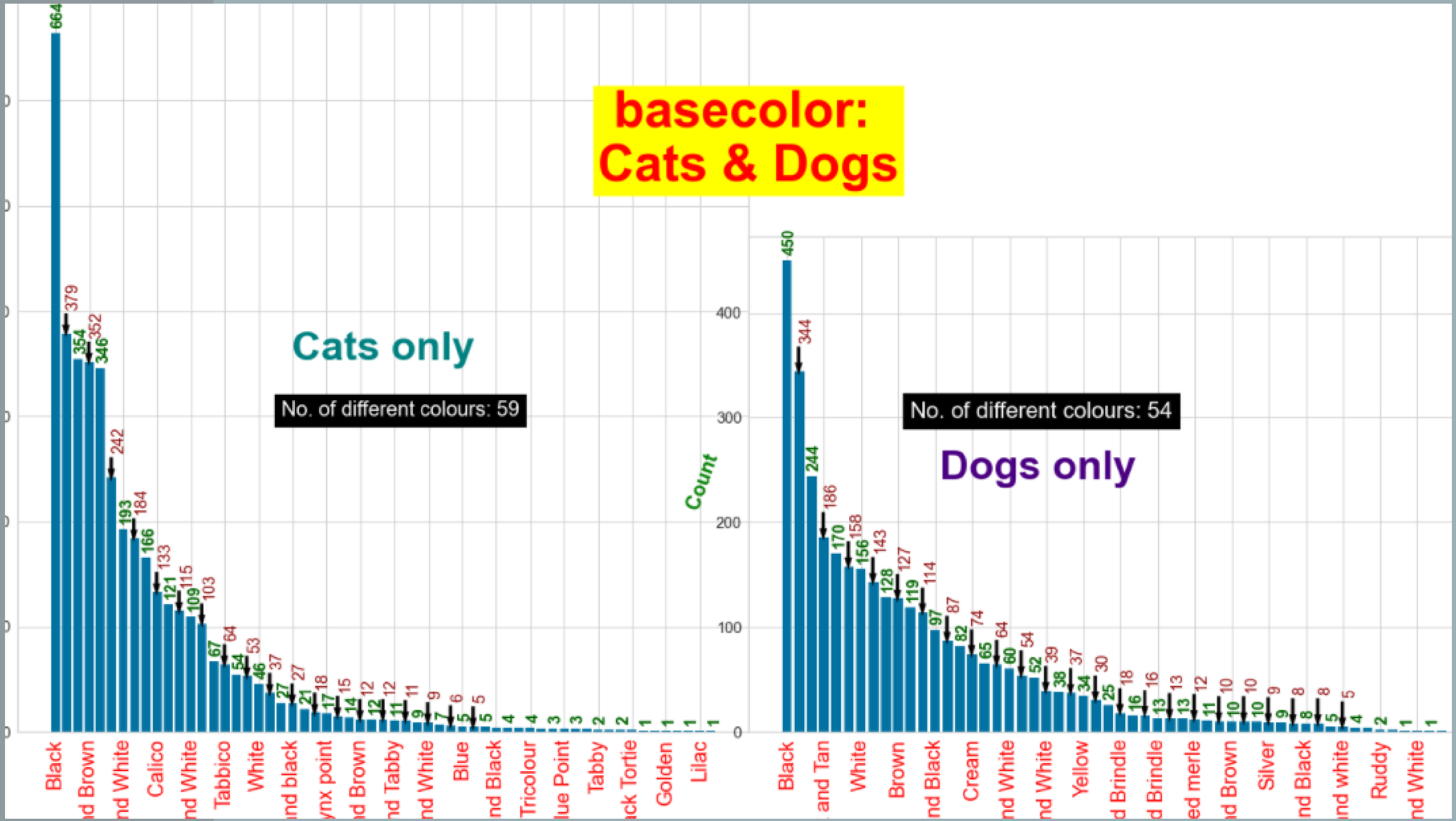


While black is the most abandoned, it si alsothe most adopted colour between cats and dogs
And strangely most shelters only have One animal very very strange



EDA: KNOWING YOUR DATA

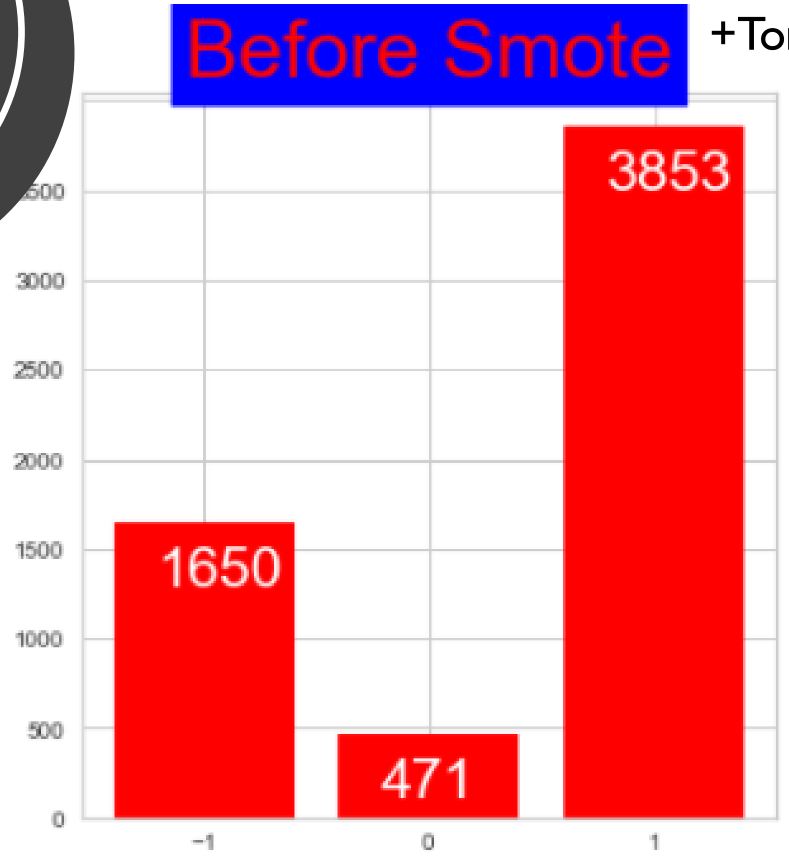
More interesting facts



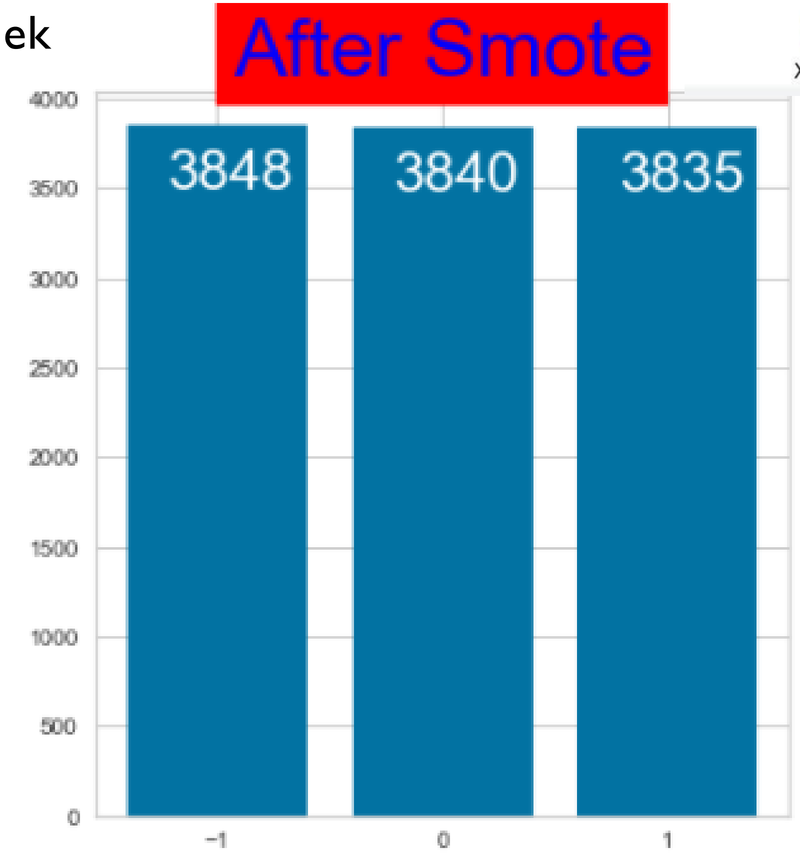
Fun fact:
More Cats are in shelters than dogs

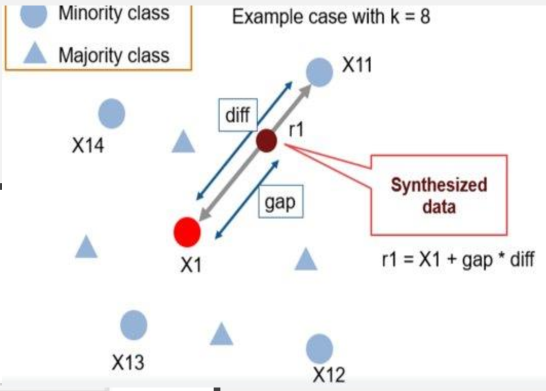
ML: LABEL

```
Before, the Counter({1: 3853, -1: 1650, 0: 471})
Class=1, n=3853 (64.496%)
Class=-1, n=1650 (27.620%)
Class=0, n=471 (7.884%)
```



+Tomek

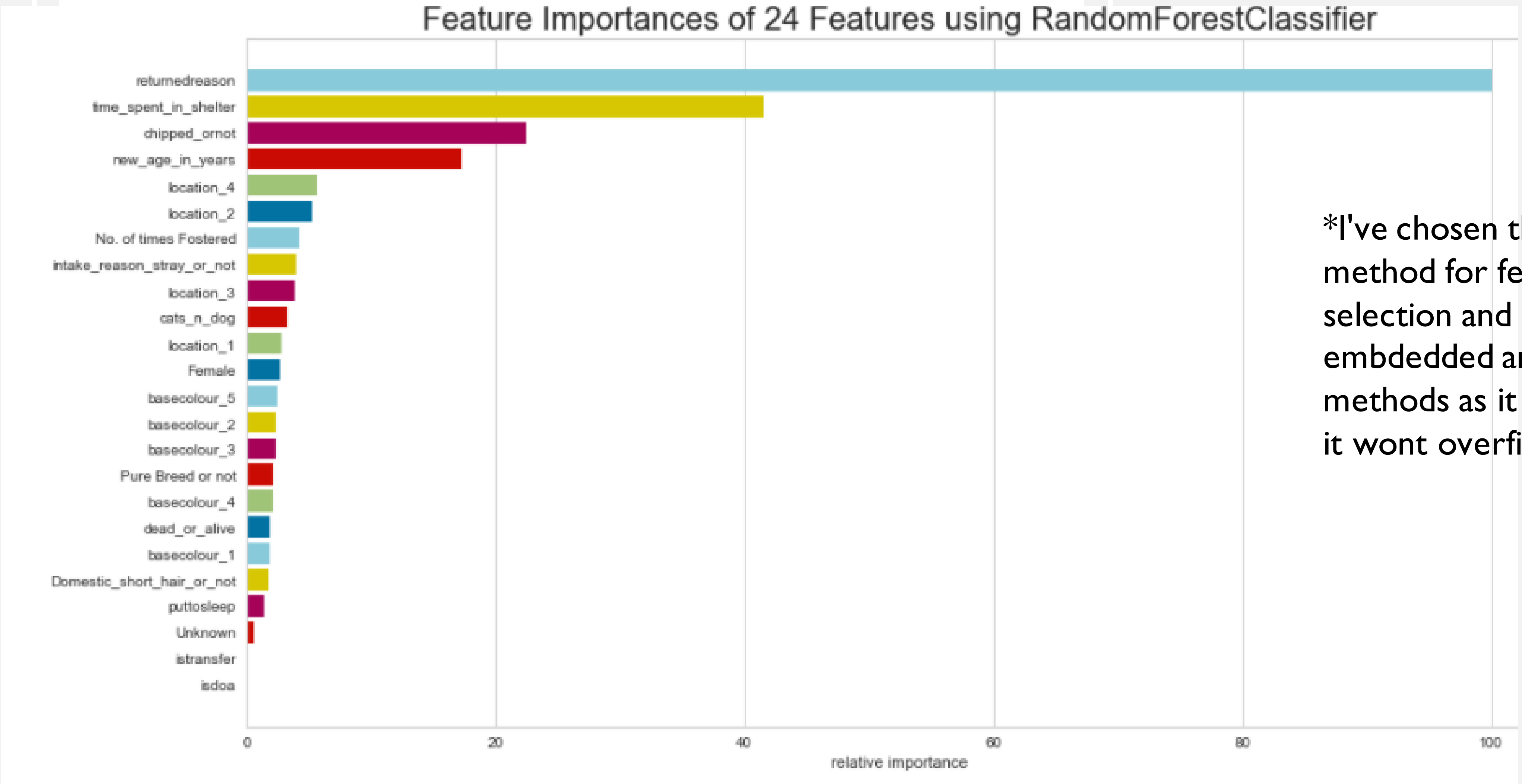





```
after, the Counter({-1: 3848, 0: 3840, 1: 3835})
Class=1, n=3835 (33.281%)
Class=-1, n=3848 (33.394%)
Class=0, n=3840 (33.325%)
```

- **How well a column can contribute to prediction of my label**

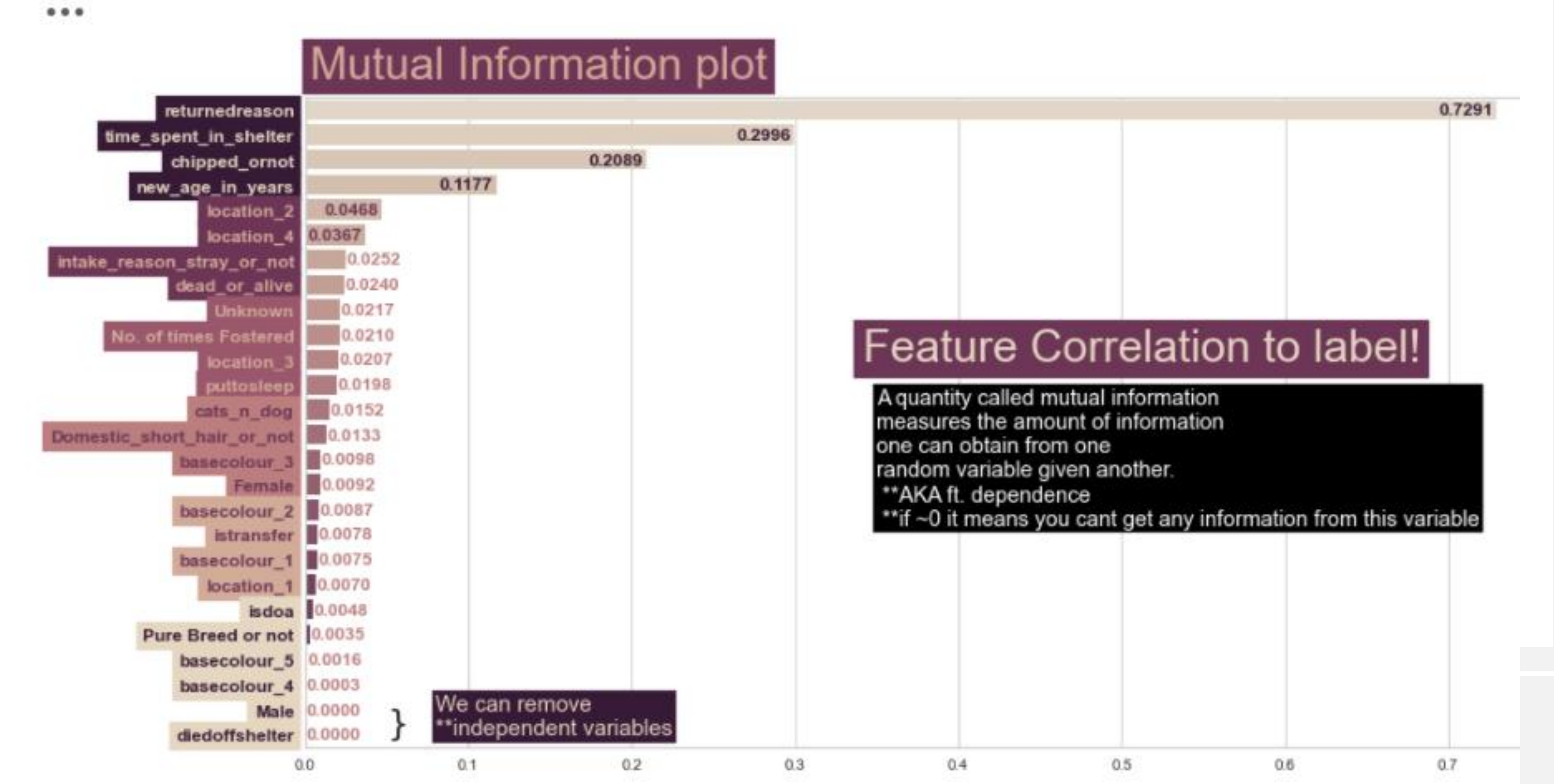
ML: FEATURE ANALYSIS



*I've chosen the filter method for feature selection and analysis over embdedded and wrapper methods as it is visual and it wont overfit as easily

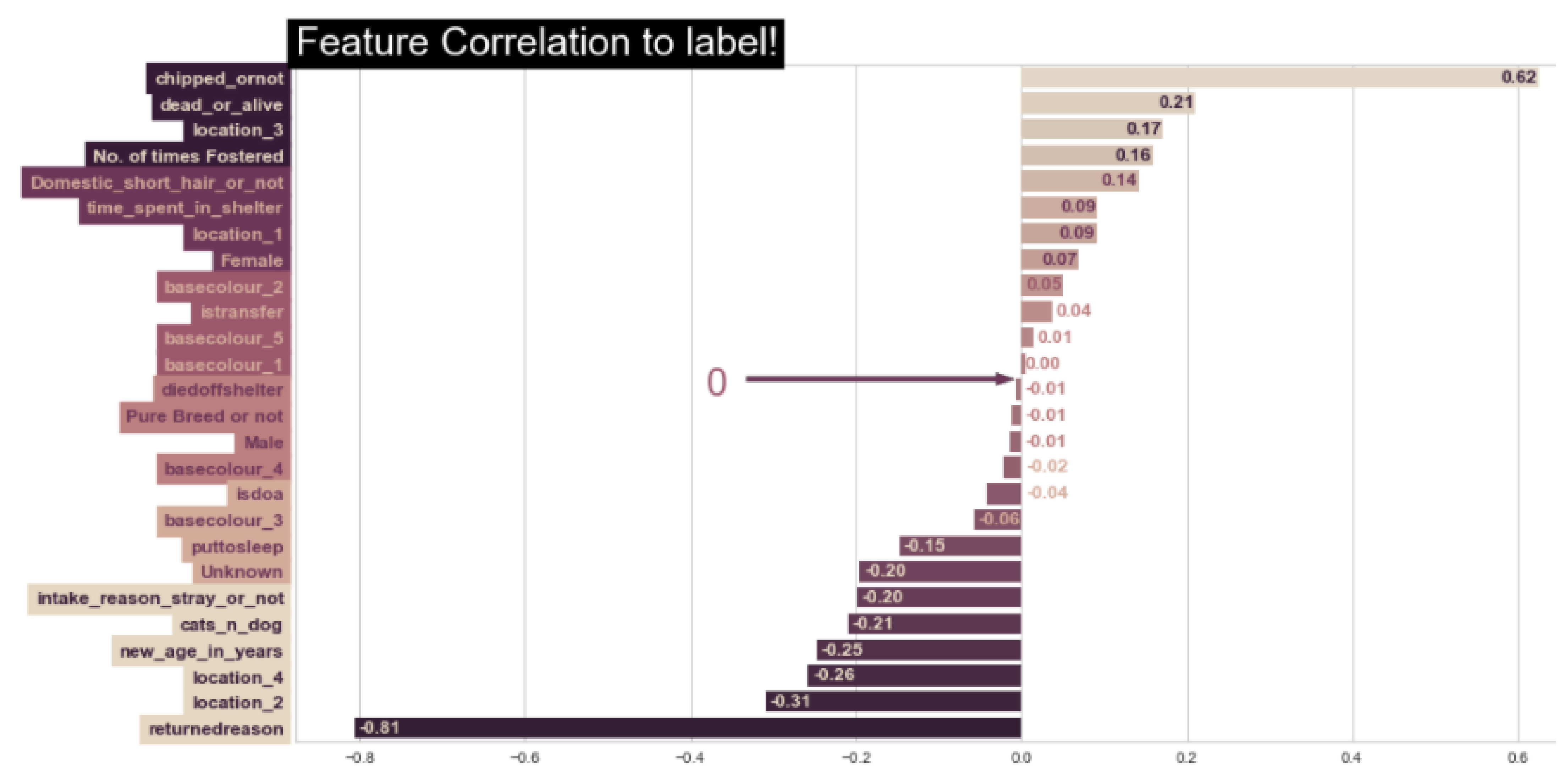
ML: FEATURE ANALYSIS

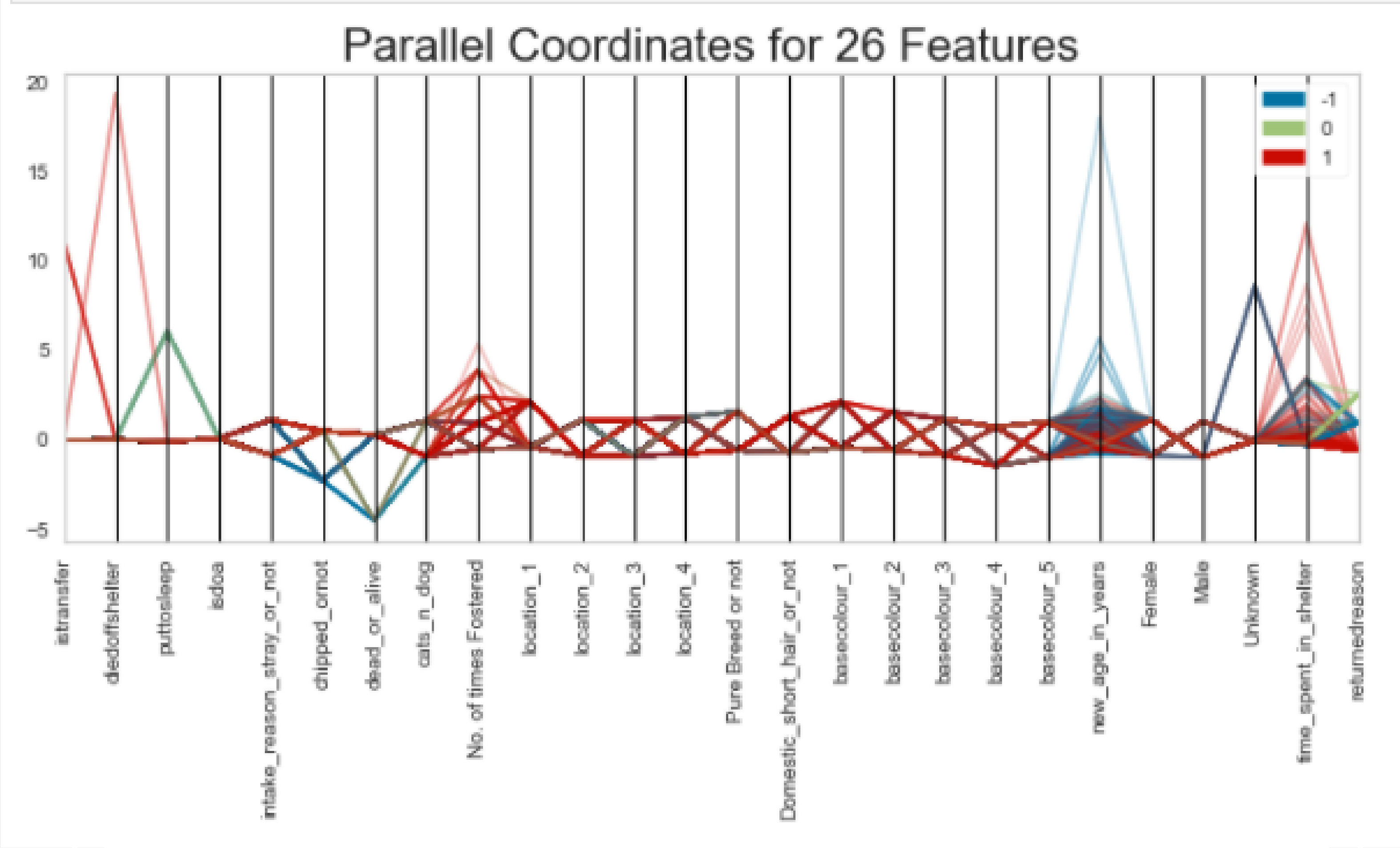
- **Chosen to drop Male & Died off shelter since no info can be obtained**



ML: FEATURE ANALYSIS

- **Highly correlated features might not impact my metrics but it can affect how I interpret the model AKA which features really played a part**





ML: FEATURE ANALYSIS

- **Can help to tell variance and if the model can even tell the difference from each class**

Model Selection

- 1. **Lazy predict** -> Suggest what models i should run
- 2. **Hyperopt** -> autoML
- 3. **TPOT** -> autoML

LazyPredictClassifier

```
100%|██████████| 29/29 [01:02<00:00,  2.16s/it]
for 'Adopted or not' multiclass-Label
```

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
XGBClassifier	0.98	0.93	None	0.98	1.09
BaggingClassifier	0.98	0.93	None	0.98	0.17
RandomForestClassifier	0.98	0.93	None	0.98	0.68
DecisionTreeClassifier	0.97	0.93	None	0.97	0.04

HyperOpt

step 3) fitting

```
hyper.fit(a_train_stand.to_numpy(),b_train.to_numpy())
```

```
100%|██████████| 1/1 [00:04<00:00,  4.54s/trial, best loss: 0.024267782426778295]
100%|██████████| 2/2 [00:09<00:00,  9.72s/trial, best loss: 0.024267782426778295]
100%|██████████| 3/3 [00:32<00:00, 32.41s/trial, best loss: 0.024267782426778295]
100%|██████████| 4/4 [00:08<00:00,  8.85s/trial, best loss: 0.024267782426778295]
100%|██████████| 5/5 [00:02<00:00,  2.55s/trial, best loss: 0.024267782426778295]
100%|██████████| 6/6 [00:04<00:00,  5.00s/trial, best loss: 0.024267782426778295]
100%|██████████| 7/7 [00:05<00:00,  5.19s/trial, best loss: 0.024267782426778295]
100%|██████████| 8/8 [00:03<00:00,  3.01s/trial, best loss: 0.023430962343096273]
100%|██████████| 9/9 [00:03<00:00,  3.90s/trial, best loss: 0.023430962343096273]
```

step 4) results

```
...
Accuracy: 0.975226
{'learner': SVC(C=43.55045088966252, cache_size=512, degree=1, gamma=0.0028828818626642524,
    max_iter=234625374.0, random_state=1, shrinking=False,
    tol=0.0015963489030502776), 'preprocs': (PCA(n_components=24),), 'ex_preprocs': ()}
```

TPOT

```
Generation 1 - Current best internal CV score: 0.9815713566763506

Generation 2 - Current best internal CV score: 0.981867915634296

Generation 3 - Current best internal CV score: 0.9819058511810427

Generation 4 - Current best internal CV score: 0.982054901453926

Generation 5 - Current best internal CV score: 0.982054901453926

Best pipeline: XGBClassifier(input_matrix, learning_rate=0.5, max_depth=1, min_child_weight=2, n_estim
]: TPOTClassifier(cv=StratifiedKFold(n_splits=10, random_state=None, shuffle=False),
               generations=5, n_jobs=4, population_size=50, random_state=42,
               scoring='roc_auc_ovr', verbosity=2)
...
roc_auc_ovr score:
]: 0.9803668502931683
```

Conclusion:

Lazy -> RF
Hyper -> SVC
TPOT -> XGB

* do note that XGBoost is not in the HyperoptEstimator library

Standardize data

- rescaling the distribution of values so that the mean of observed
- centering the data.

```
# https://towardsdatascience.com/what-and-why-behind-fit-tr
 # -> Standardize -> rescaling the distribution of values s
from sklearn.preprocessing import StandardScaler
num_cols = ["No. of times Fostered","new_age_in_years","tim
a_train_stand = a_train.copy() #convert array to DF
a_test_stand = a_test.copy()
# apply standardization on numerical features
for i in num_cols:
    # fit        on training data column
    sc    = StandardScaler().fit(a_train_stand[[i]])

    # transform the training data column
    a_train_stand[i] = sc.transform(a_train_stand[[i]])

    # transform the testing data column
    a_test_stand[i] = sc.transform(a_test_stand[[i]])
```

Normalize data

```
# range between 0 to 1. So, normalization would not affect
# fit scaler on training data
norm = MinMaxScaler().fit(a_train)

# transform training data
a_train_normz = norm.transform(a_train)

# transform testing dataabs
a_test_normz = norm.transform(a_test)
```

Not scaled, Standardized & Normalized data

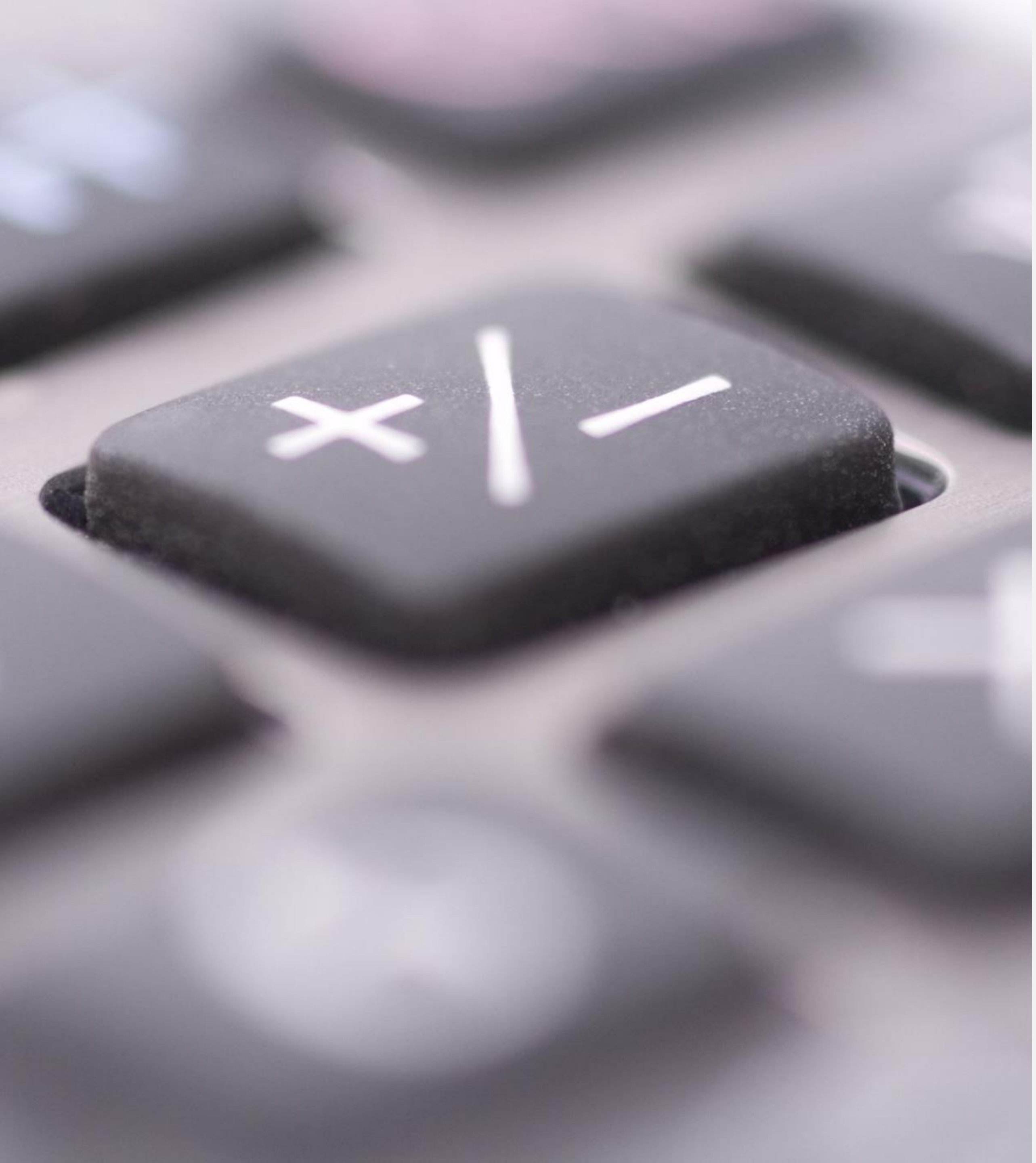
```
traina = [a_train, a_train_normz, a_train_stand]
testa  = [a_test,   a_test_normz, a_test_stand]
```

ML: DATA SCALE

- **I've chosen to keep a set of 3 to see if my dataset is sensitive to any 3 of these different scales of data**

Normalization typically means rescales the values into a range of [0,1].

Standardization typically means rescales data to have a mean of 0 and a **standard** deviation of 1



ML: METRIC REVISION

- Accuracy -> Correctness
-     - (tp + tn) / (p + n)
- Precision -> Exactness
-     - tp / (tp + fp)
- Recall -> Completeness
-     - tp / (tp + fn)
- F1 -> how complete & exact IE balance of precision and recall
-     - 2 tp / (2 tp + fp + fn)
- Logloss -> model comparison metric
- AUC ROC ovo and ovr:
- Ovr is harder to determine the difference between classes for many classes and is used typically for a faster metric
- If possible ovo is the better choice

```
Not scaled!!!!!!!!!!!!
number of train sample in train set: (5974, 24)
Number of samples in validation set: (1494,)
TRAINing with RF.score: 100.0
Log Loss:                  0.10243527626924914 * The lower the better.
ROC AUC (One-vs-one) :     0.9791774040521516  * The higher the better.
ROC AUC (One-vs-rest):     0.9830746076943381  * same ^
-----------------------------------------------------------------
              precision    recall  f1-score   support

          -1       0.98      0.98      0.98       408
           0       0.93      0.81      0.86       125
           1       0.98      1.00      0.99       961

    accuracy                           0.98      1494
   macro avg       0.96      0.93      0.94      1494
weighted avg       0.98      0.98      0.98      1494
```

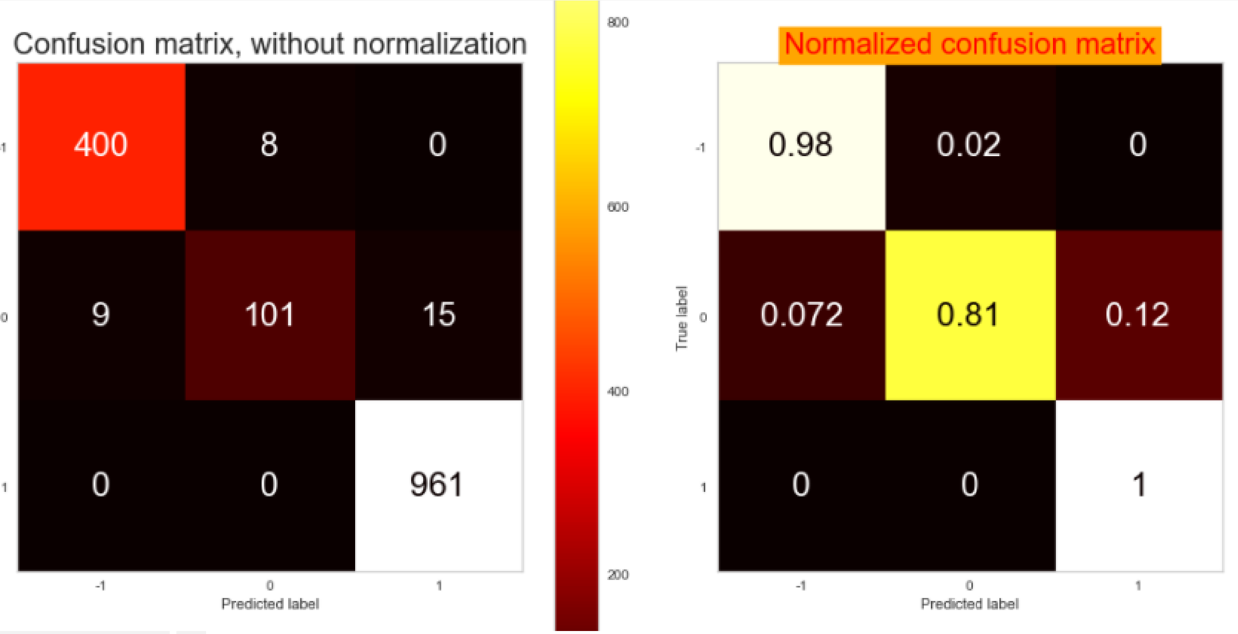
```
Standardised!!!!!!!!!!
number of train sample in train set: (5974, 24)
Number of samples in validation set: (1494,)
TRAINing with RF.score: 100.0
Log Loss:                  0.10246268026918709 * The lower the better.
ROC AUC (One-vs-one) :     0.9791895526854516  * The higher the better.
ROC AUC (One-vs-rest):     0.9830782553728911  * same ^
-----------------------------------------------------------------
              precision    recall  f1-score   support

          -1       0.98      0.98      0.98       408
           0       0.93      0.81      0.86       125
           1       0.98      1.00      0.99       961

    accuracy                           0.98      1494
   macro avg       0.96      0.93      0.94      1494
weighted avg       0.98      0.98      0.98      1494
```

```
Normalised!!!!!!!!!!!!
number of train sample in train set: (5974, 24)
Number of samples in validation set: (1494,)
TRAINing with RF.score: 100.0
Log Loss:                  0.1021840561354754 * The lower the better.
ROC AUC (One-vs-one) :     0.9793772707487435  * The higher the better.
ROC AUC (One-vs-rest):     0.9832990755387098  * same ^
-----------------------------------------------------------------
              precision    recall  f1-score   support

          -1       0.98      0.98      0.98       408
           0       0.93      0.81      0.86       125
           1       0.98      1.00      0.99       961

    accuracy                           0.98      1494
   macro avg       0.96      0.93      0.94      1494
weighted avg       0.98      0.98      0.98      1494

-----------------------------------------------------------------
```

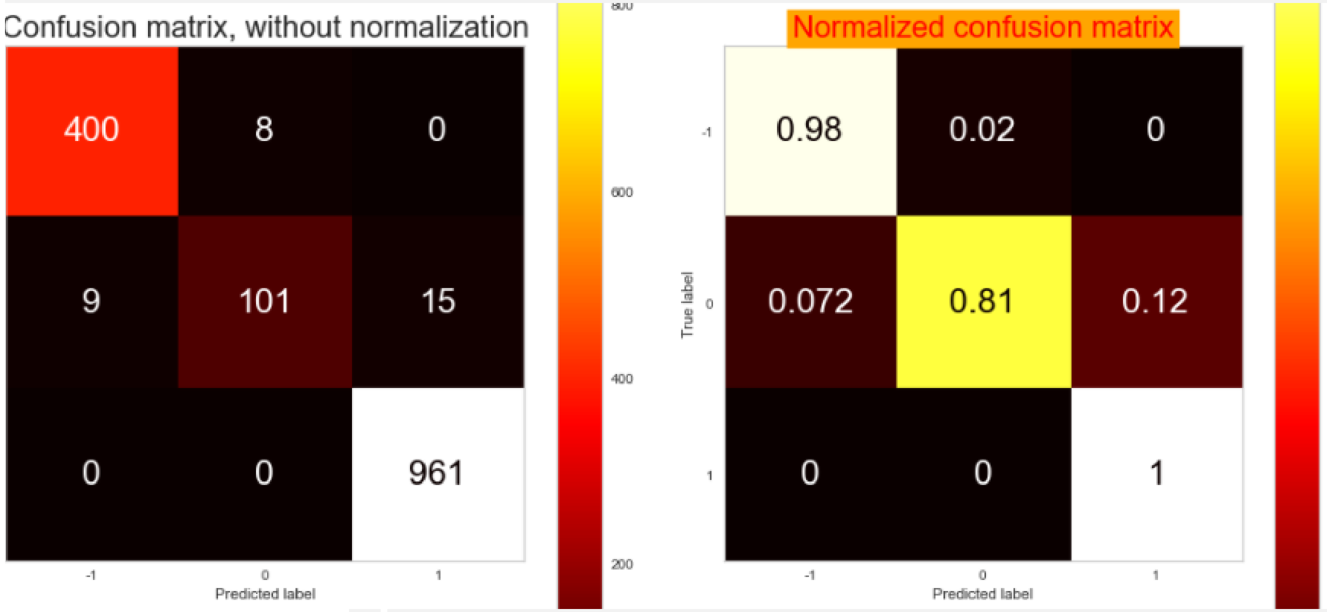
```
SmoteTomek!!!!!!!!!!
number of train sample in train set: (11523, 24)
Number of samples in validation set: (1494,)
TRAINing with RF.score: 100.0
Log Loss:                  0.14091266293948082 * The lower the better.
ROC AUC (One-vs-one) :     0.9803006280902927  * The higher the better.
ROC AUC (One-vs-rest):     0.9842574384231834  * same ^
-----------------------------------------------------------------
              precision    recall  f1-score   support

          -1       0.99      0.94      0.97       408
           0       0.78      0.85      0.81       125
           1       0.98      0.99      0.99       961

    accuracy                           0.97      1494
   macro avg       0.92      0.93      0.92      1494
weighted avg       0.97      0.97      0.97      1494
```

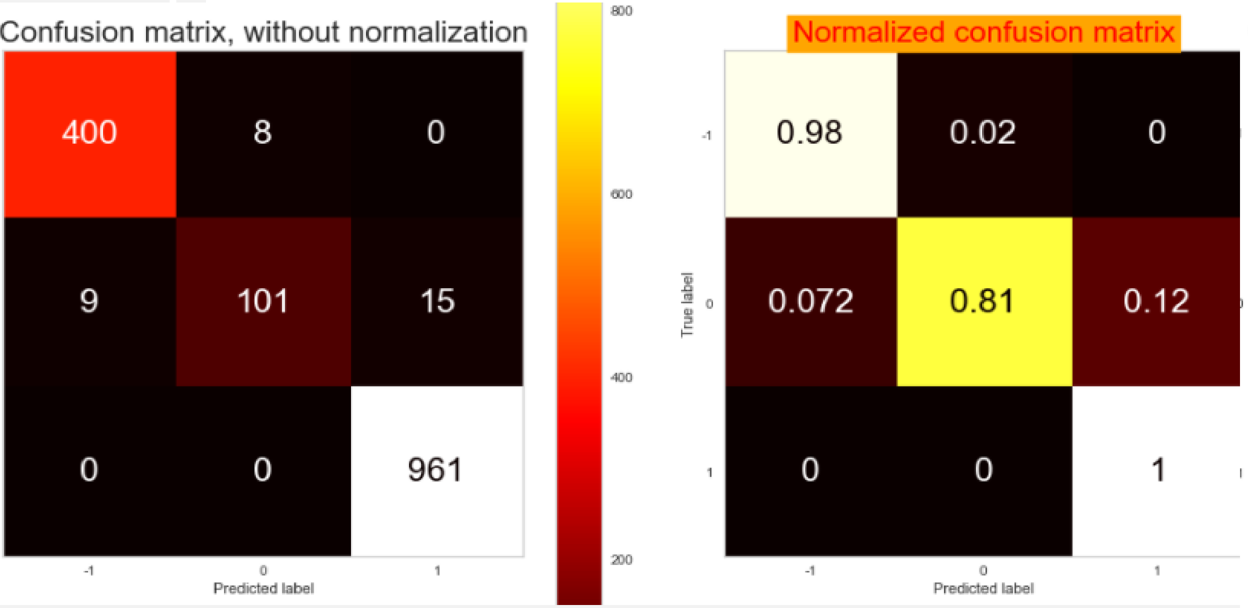
ML: RANDOMFOREST MODELS & METRICS

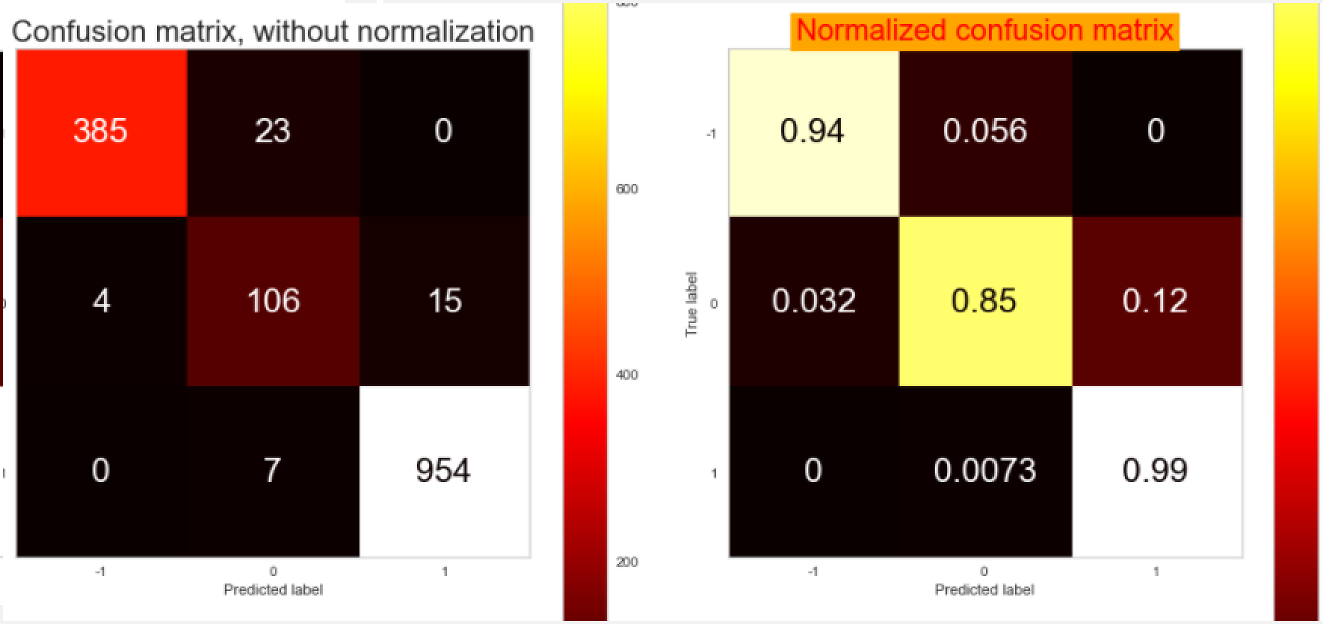
Non scaled



Standardised







Standardised

For standardised (since score is very similar)

Hyperparameter tuning

#1 Grid Search~~

```
base_rf = RandomForestClassifier(random_state = 42, class_weight = "balanced")
```

```
param_dict6 = { 'max_depth' : np.arange(10,20) ,
                'criterion': ['gini','entropy'],
                'n_estimators': np.arange(900,1000,5)}
grid_model = GridSearchCV(estimator= base_rf, param_grid = param_dict6 , verbose= 1,n_jobs=16, refit = True, \
                             cv=RepeatedStratifiedKFold(n_splits=4,n_repeats=3,random_state=42))
grid_model.fit(a_train_stand, b_train)
grid_model.best_params_
```

```
Fitting 12 folds for each of 400 candidates, totalling 4800 fits
[Parallel(n_jobs=16)]: Using backend LokyBackend with 16 concurrent workers.
[Parallel(n_jobs=16)]: Done  18 tasks      | elapsed:   21.7s
[Parallel(n_jobs=16)]: Done 168 tasks      | elapsed:  3.0min
[Parallel(n_jobs=16)]: Done 418 tasks      | elapsed:  8.0min
[Parallel(n_jobs=16)]: Done 768 tasks      | elapsed: 15.6min
[Parallel(n_jobs=16)]: Done 1218 tasks      | elapsed: 25.8min
[Parallel(n_jobs=16)]: Done 1768 tasks      | elapsed: 38.3min
[Parallel(n_jobs=16)]: Done 2418 tasks      | elapsed: 53.0min
[Parallel(n_jobs=16)]: Done 3168 tasks      | elapsed: 70.2min
[Parallel(n_jobs=16)]: Done 4018 tasks      | elapsed: 90.1min
[Parallel(n_jobs=16)]: Done 4800 out of 4800 | elapsed: 108.7min finished
```

```
{'criterion': 'gini', 'max_depth': 16, 'n_estimators': 975}
```

```
grid_model.best_estimator_
```

```
RandomForestClassifier(class_weight='balanced', max_depth=16, n_estimators=975,
                       random_state=42)
```

For smoted

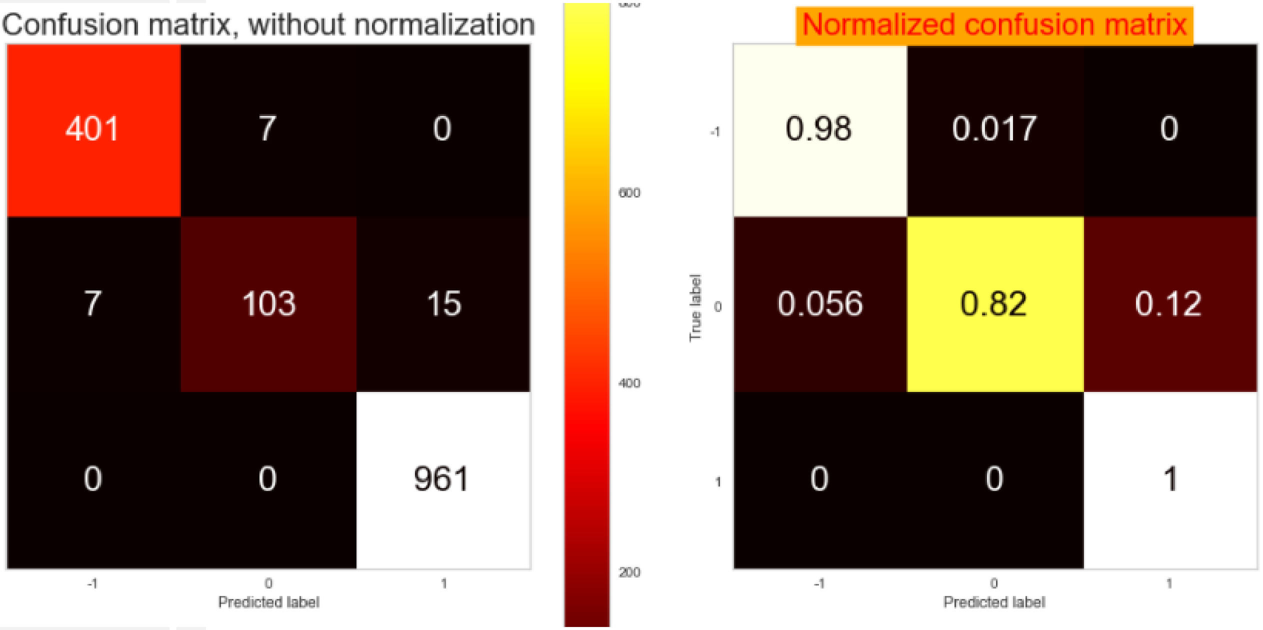
#2 Optuna

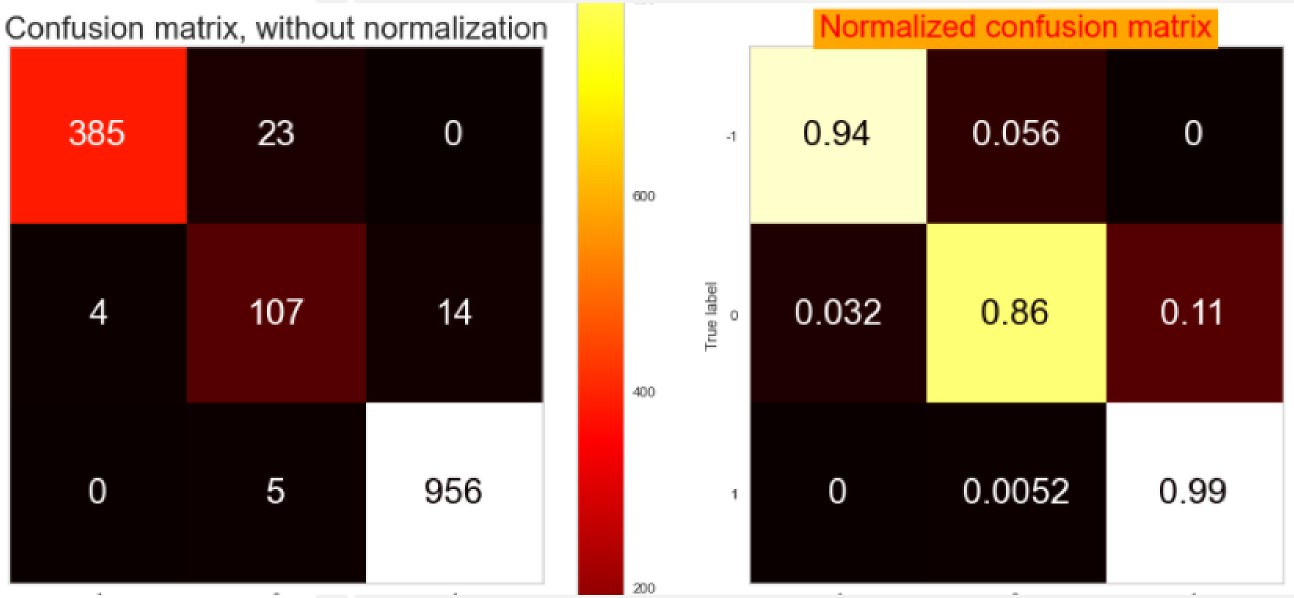
...

```
[I 2021-06-12 05:43:08,928] A new study created in memory with name: no-name-22ce870e-221e-40f7-9e49-6114f0a71898
[I 2021-06-12 05:43:10,063] Trial 0 finished with value: 0.30035779897337495 and parameters: {'n_estimators': 128, 'max_depth': 6.
[I 2021-06-12 05:43:16,595] Trial 1 finished with value: 0.1418853989249978 and parameters: {'n_estimators': 583, 'max_depth': 29.
[I 2021-06-12 05:43:21,194] Trial 2 finished with value: 0.830471663608147 and parameters: {'n_estimators': 759, 'max_depth': 1.06
[I 2021-06-12 05:43:22,142] Trial 3 finished with value: 0.3674493673077322 and parameters: {'n_estimators': 94, 'max_depth': 5.71
[I 2021-06-12 05:43:30,343] Trial 4 finished with value: 0.22972692067624803 and parameters: {'n_estimators': 868, 'max_depth': 8.
[I 2021-06-12 05:43:30,740] Trial 5 finished with value: 0.3792928367736469 and parameters: {'n_estimators': 46, 'max_depth': 5.04
[I 2021-06-12 05:43:42,954] Trial 6 finished with value: 0.14120368378076442 and parameters: {'n_estimators': 963, 'max_depth': 24
[I 2021-06-12 05:43:50,035] Trial 7 finished with value: 0.14162992751967415 and parameters: {'n_estimators': 559, 'max_depth': 20
[I 2021-06-12 05:43:54,661] Trial 8 finished with value: 0.8326376623232614 and parameters: {'n_estimators': 730, 'max_depth': 1.8
```

Standardised grid:
Trade between accuracy
over label –1 and 0

Smote optuna:
Trade between
accuracy over label 0 and 1






```
TRAINing with RF.score: 99.98
Log Loss:                   0.10361275017913892  * The lower the better.
ROC AUC (One-vs-one) :      0.9817281494630458   * The higher the better.
ROC AUC (One-vs-rest):      0.9850405687251181   * same ^
------------------------------------------------------------------
              precision    recall  f1-score   support

          -1       0.98      0.98      0.98       408
           0       0.94      0.82      0.88       125
           1       0.98      1.00      0.99       961

    accuracy                           0.98      1494
   macro avg       0.97      0.94      0.95      1494
weighted avg       0.98      0.98      0.98      1494
```

```
SmoteTomek tuned!!!!!!!!!!
number of train sample in train set: (11523, 24)
Number of samples in validation set: (1494,)
TRAINing with RF.score: 100.0
Log Loss:                   0.1211218136306414  * The lower the better.
ROC AUC (One-vs-one) :      0.9816682496446375  * The higher the better.
ROC AUC (One-vs-rest):      0.9853510580290917  * same ^
------------------------------------------------------------------
              precision    recall  f1-score   support

          -1       0.99      0.94      0.97       408
           0       0.79      0.86      0.82       125
           1       0.99      0.99      0.99       961

    accuracy                           0.97      1494
   macro avg       0.92      0.93      0.93      1494
weighted avg       0.97      0.97      0.97      1494
```

```
Not scaled!!!!!!!!!!!!
number of train sample in train set: (5974, 24)
Number of samples in validation set: (1494,)
TRAINing with RF.score: 97.49
Log Loss:                0.11953568473968472  * The lower the better.
ROC AUC (One-vs-one) :   0.9634357559187393   * The higher the better.
ROC AUC (One-vs-rest):   0.9747612472765371   * same ^
-----------------------------------------------------------------
              precision    recall  f1-score   support

          -1       0.96      1.00      0.98       408
           0       1.00      0.75      0.86       125
           1       0.98      1.00      0.99       961

    accuracy                           0.98      1494
   macro avg       0.98      0.92      0.94      1494
weighted avg       0.98      0.98      0.98      1494
```

```
Standardised!!!!!!!!!!
number of train sample in train set: (5974, 24)
Number of samples in validation set: (1494,)
TRAINing with RF.score: 97.52
Log Loss:                0.11802444817263633  * The lower the better.
ROC AUC (One-vs-one) :   0.9642616589473111   * The higher the better.
ROC AUC (One-vs-rest):   0.975346998890427   * same ^
-----------------------------------------------------------------
              precision    recall  f1-score   support

          -1       0.96      1.00      0.98       408
           0       0.99      0.75      0.85       125
           1       0.98      1.00      0.99       961

    accuracy                           0.98      1494
   macro avg       0.98      0.92      0.94      1494
weighted avg       0.98      0.98      0.98      1494
```

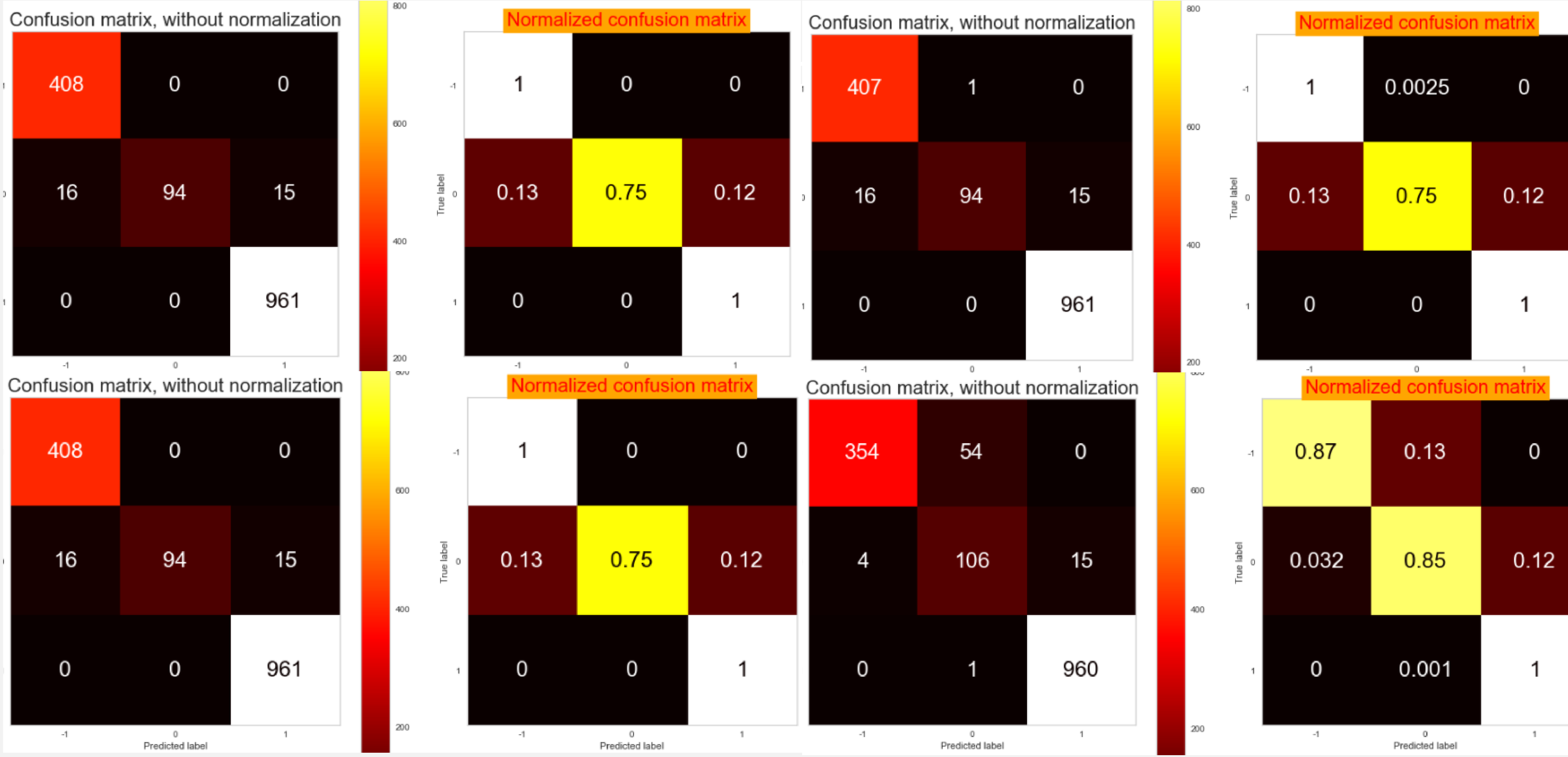
```
Normalised!!!!!!!!!!!!
number of train sample in train set: (5974, 24)
Number of samples in validation set: (1494,)
TRAINing with RF.score: 97.47
Log Loss:                0.12752044578648455  * The lower the better
ROC AUC (One-vs-one) :   0.9650315915474758   * The higher the bette
ROC AUC (One-vs-rest):   0.9755247894164777   * same ^
-----------------------------------------------------------------
              precision    recall  f1-score   support

          -1       0.96      1.00      0.98       408
           0       1.00      0.75      0.86       125
           1       0.98      1.00      0.99       961

    accuracy                           0.98      1494
   macro avg       0.98      0.92      0.94      1494
weighted avg       0.98      0.98      0.98      1494
```

```
SmoteTomek tuned!!!!!!!!!!
number of train sample in train set: (11523, 24)
Number of samples in validation set: (1494,)
TRAINing with RF.score: 91.83
Log Loss:                0.17615771417689263  * The lower the better.
ROC AUC (One-vs-one) :   0.9654515754966573   * The higher the better.
ROC AUC (One-vs-rest):   0.9740726148722928   * same ^
-----------------------------------------------------------------
              precision    recall  f1-score   support

          -1       0.99      0.87      0.92       408
           0       0.66      0.85      0.74       125
           1       0.98      1.00      0.99       961

    accuracy                           0.95      1494
   macro avg       0.88      0.90      0.89      1494
weighted avg       0.96      0.95      0.95      1494
```

ML: SVC MODELS & METRICS

Non scaled

Standardised



Standardised

SVC was already tuned with the HyperOpt autoML! Bam!

ML: SVC
MODELS & METRICS

```
Not scaled!!!!!!!!!!!!
number of train sample in train set: (5974, 24)
Number of samples in validation set: (1494,)
TRAINing with RF.score: 97.71
Log Loss:                 0.0791700865157313  * The lower the better.
ROC AUC (One-vs-one) :    0.9765716097746765  * The higher the better.
ROC AUC (One-vs-rest):    0.9803668502931683  * same ^
-------------------------------------------------------------------
              precision    recall  f1-score   support

          -1       0.97      0.98      0.98       408
           0       0.92      0.79      0.85       125
           1       0.98      1.00      0.99       961

    accuracy                           0.98      1494
   macro avg       0.96      0.92      0.94      1494
weighted avg       0.98      0.98      0.98      1494
```

```
Standardised!!!!!!!!!!
number of train sample in train set: (5974, 24)
Number of samples in validation set: (1494,)
TRAINing with RF.score: 97.71
Log Loss:                 0.0791700865157313  * The lower the better.
ROC AUC (One-vs-one) :    0.9765716097746765  * The higher the better.
ROC AUC (One-vs-rest):    0.9803668502931683  * same ^
-------------------------------------------------------------------
              precision    recall  f1-score   support

          -1       0.97      0.98      0.98       408
           0       0.92      0.79      0.85       125
           1       0.98      1.00      0.99       961

    accuracy                           0.98      1494
   macro avg       0.96      0.92      0.94      1494
weighted avg       0.98      0.98      0.98      1494
```

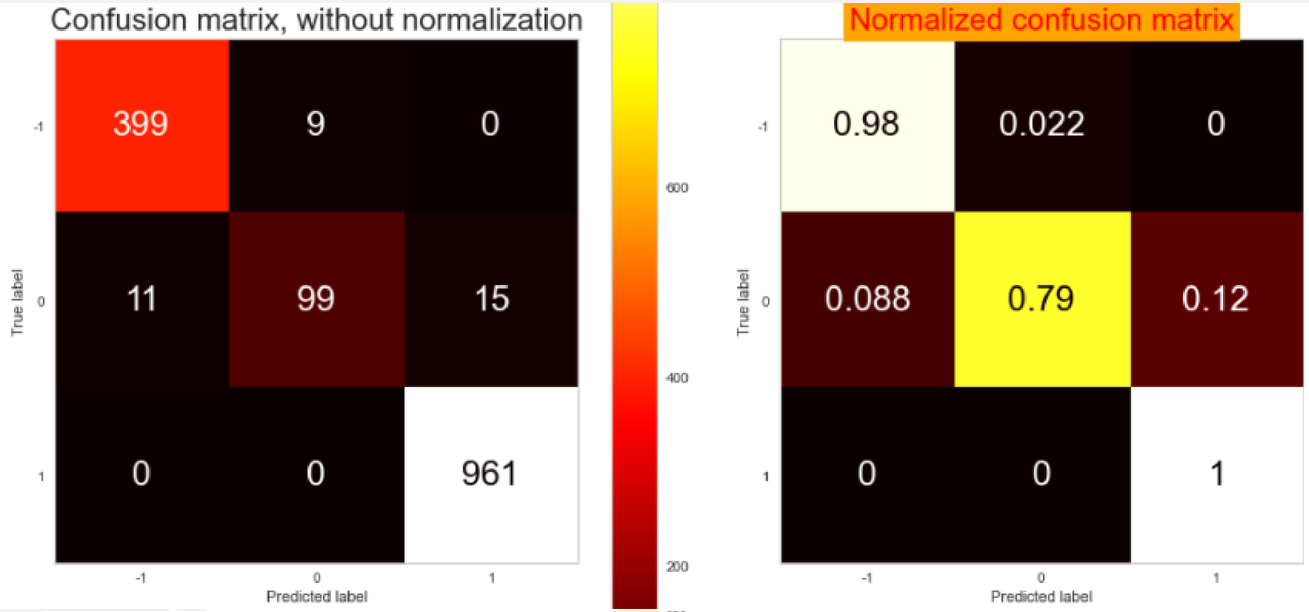
```
Normalised!!!!!!!!!!!!
number of train sample in train set: (5974, 24)
Number of samples in validation set: (1494,)
TRAINing with RF.score: 97.71
Log Loss:                 0.0791700865157313  * The lower the better.
ROC AUC (One-vs-one) :    0.9765716097746765  * The higher the better.
ROC AUC (One-vs-rest):    0.9803668502931683  * same ^
-------------------------------------------------------------------
              precision    recall  f1-score   support

          -1       0.97      0.98      0.98       408
           0       0.92      0.79      0.85       125
           1       0.98      1.00      0.99       961

    accuracy                           0.98      1494
   macro avg       0.96      0.92      0.94      1494
weighted avg       0.98      0.98      0.98      1494
```

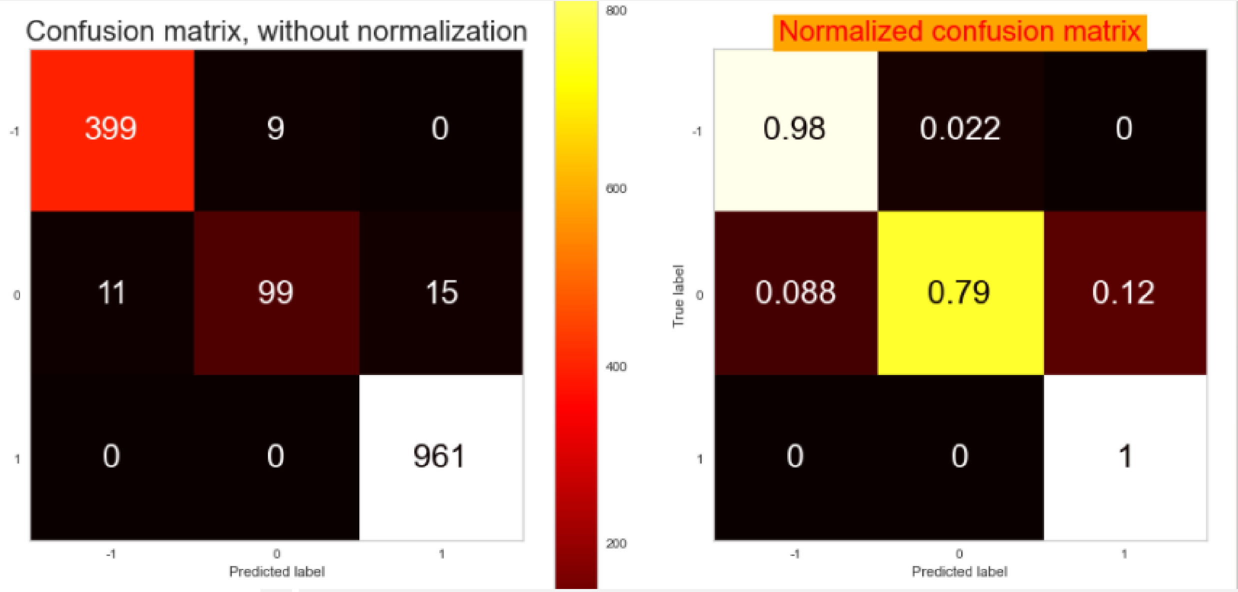
```
SmoteTomek tuned!!!!!!!!!!
number of train sample in train set: (11523, 24)
Number of samples in validation set: (1494,)
TRAINing with RF.score: 94.03
Log Loss:                 0.13679532512239795  * The lower the better.
ROC AUC (One-vs-one) :    0.973639091224419    * The higher the better.
ROC AUC (One-vs-rest):    0.9776815108827893   * same ^
-------------------------------------------------------------------
              precision    recall  f1-score   support

          -1       0.99      0.94      0.96       408
           0       0.80      0.84      0.82       125
           1       0.98      1.00      0.99       961

    accuracy                           0.97      1494
   macro avg       0.92      0.93      0.93      1494
weighted avg       0.97      0.97      0.97      1494
```

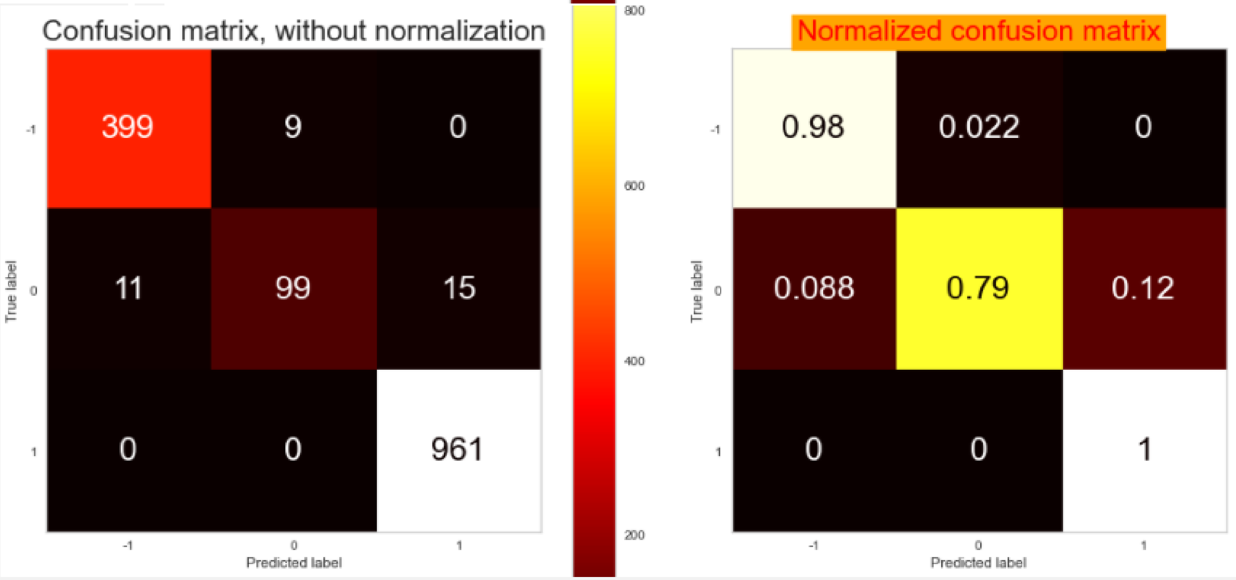
ML: XGBOOST MODELS & METRICS

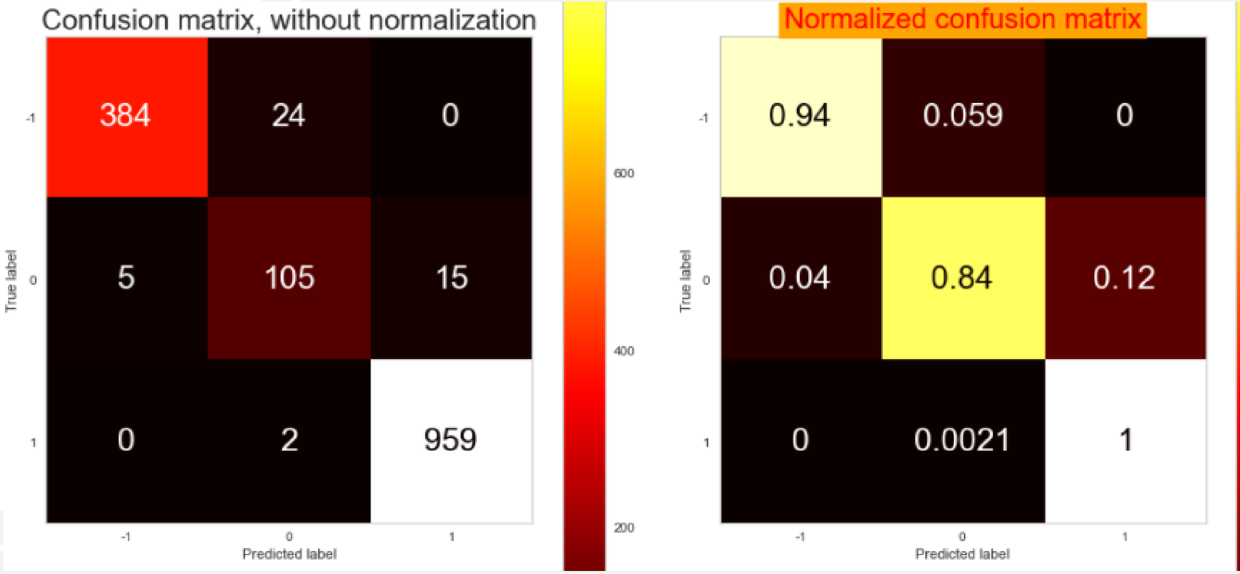
Non scaled

Standardised





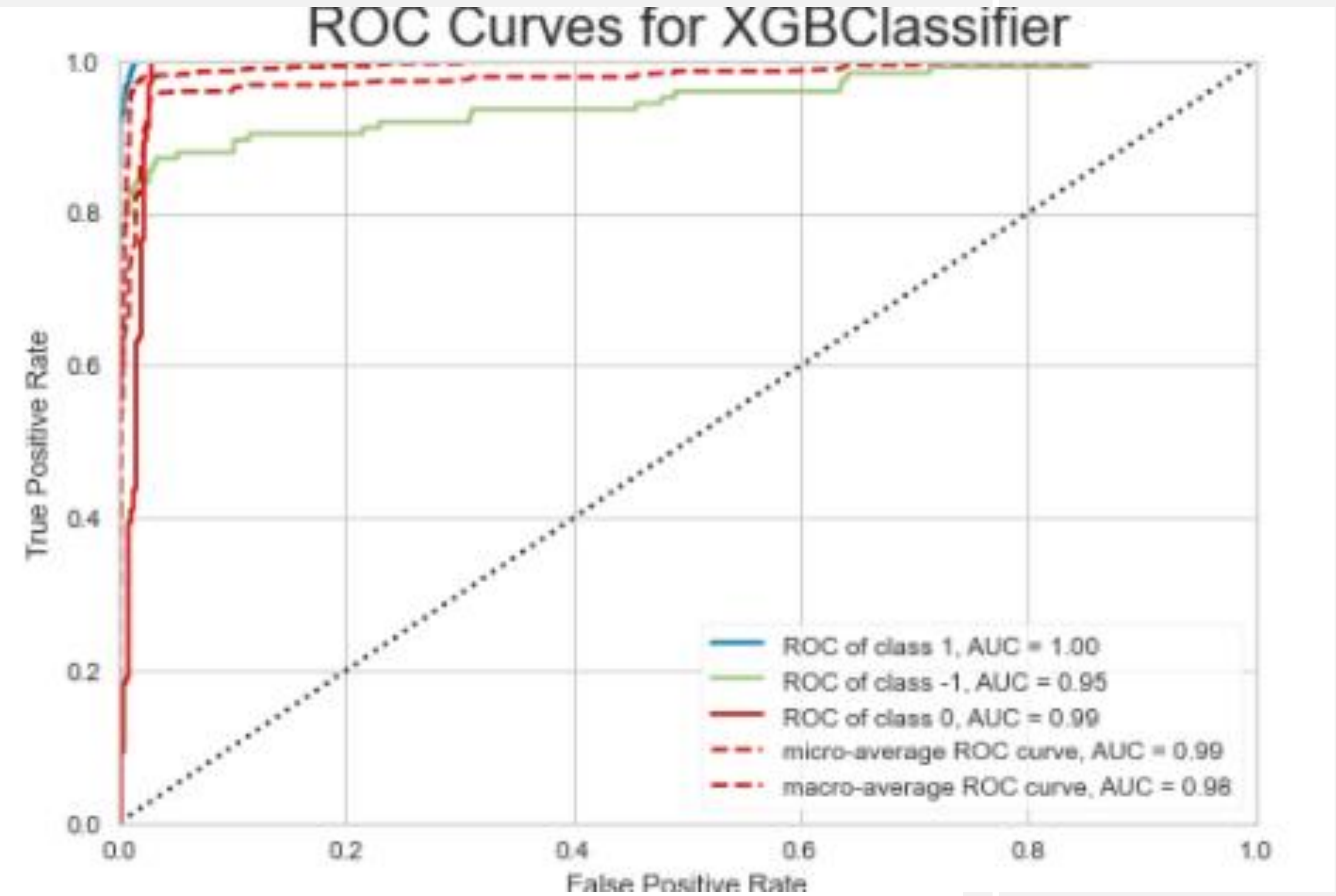




Standardised

XGB was already tuned with the TPOT autoML! Bam!

ML: XGBOOST
MODELS & METRICS



ML: XGBOOST
MODELS & METRICS

ATTEMPTING AN NN

```
# compile the keras model
from keras.optimizers import SGD
model = Sequential()
model.add(Dense(12, input_dim=24, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='categorical_crossentropy', optimizer="adam", metrics=['accuracy'])
model.fit(ker, kerr, epochs=150, batch_size=10)
_, accuracy = model.evaluate(ker, kerr)
print('Accuracy: %.2f' % (accuracy*100))
```

```
Epoch 1/150
1153/1153 [==============================] - 4s 555us/step - loss: 0.0000e+00 - accuracy: 0.3425
Epoch 2/150
1153/1153 [==============================] - 1s 544us/step - loss: 0.0000e+00 - accuracy: 0.3364
Epoch 3/150
1153/1153 [==============================] - 1s 548us/step - loss: 0.0000e+00 - accuracy: 0.3448
Epoch 4/150
1153/1153 [==============================] - 1s 584us/step - loss: 0.0000e+00 - accuracy: 0.3331
Epoch 5/150
```

```
Epoch 150/150
1153/1153 [==============================] - 1s 1ms/s
361/361 [==============================] - 0s 740us/s
Accuracy: 33.32
```

What a horrible score!

- One's failure today will be a victory some other day


WHOEVER SAID
MONEY CAN'T BUY HAPPINESS
HAS NEVER PAID AN ADOPTION FEE

THANK YOU!