# Kaggle Project
# Predicting Life Satisfaction

Team: Top of the CLASSifier
Sabrina Tan and Dylan Lee

# Data Cleaning & Feature Engineering

The dataset consists of 270 columns and 30,000 rows for the training set / 9,200 rows for the test set. Steps we took to clean and reduce the size of the data are:

1. Convert country responses to corresponding continent
2. Group languages
3. Smart impute ages; bin them by quantile
4. Bin several other features
5. Manually process features that have more than 30% of responses missing case by case
6. Ensure feature data type is correct (ordinal, nominal, numeric)

# Data Cleaning & Feature Engineering

We found that using a ‘predicted age’ to replace missing values for the ‘age’ feature performed better when fitting a lightGBM model than imputing missing values with the mean age. Examples of features used to predict age are:

- v54 - “Doing last 7 days: education” negatively correlated with age
- v217 - “Doing last 7 days: retired” positively correlated with age
- v218 - “Partner doing last 7 days: retired” positively correlated with age

# Data Cleaning & Feature Engineering

Considering responses that have ‘.’ as a missing response, we found that there are 71 features that have over 30% missing responses.

- Majority of these features are questions asking about the 3+ person in the household, so we decided to drop these features
- The processed data categorized missing responses as ‘0’ but there were still questions where the given responses seemed like they should be ‘0’ but weren’t, so we categorized as ‘-1’

| Question | Percentage Missing | Label |
|---|---|---|
| v207 | 99.996676 | Thirteenth person in household... |
| v89 | 99.996676 | Gender of thirteenth person in... |
| v195 | 99.996676 | Thirteenth person in household... |
| v262 | 99.996676 | Year of birth of thirteenth pe... |
| v261 | 99.983378 | Year of birth of twelfth perso... |
| v88 | 99.983378 | Gender of twelfth person in ho... |
| v206 | 99.980053 | Twelfth person in household: r... |
| v194 | 99.980053 | Twelfth person in household: r... |
| v260 | 99.940160 | Year of birth of eleventh pers... |
| v87 | 99.933511 | Gender of eleventh person in h... |
| v205 | 99.930186 | Eleventh person in household: ... |
| v193 | 99.930186 | Eleventh person in household: ... |
| v259 | 99.893617 | Year of birth of tenth person ... |
| v86 | 99.883644 | Gender of tenth person in hous... |

# Input Models

# Models - Logistic Regression

*Logistic regression is an easy model to implement and is appropriate for classification of 2 classes (satisfied / unsatisfied).*

- We used a 80/20 train/test split
- Initial run using default parameters scored AUC of 59.60

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

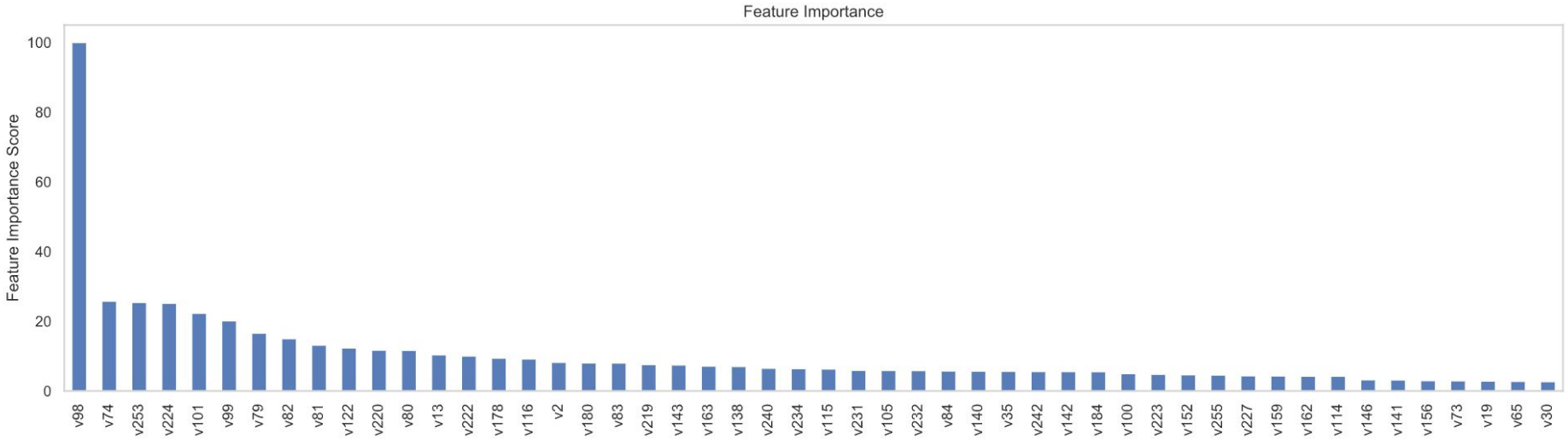
# Models - Logistic Regression

We also implemented a recursive feature elimination process to see if we could find a reduced model that performs better than the base model with all parameters

- Each feature is assigned a weight of importance and the least important features are pruned from the current model, recursively
- Then the model with the highest accuracy is chosen

- Optimum number of features: 107
- Tuned AUC 79.86%
- Submission AUC 77.35%

# Models - Logistic Regression




- Top 5: v98 (How happy are you), v74 (Enjoyed life, how often past week), v253 (Were happy, how often past week), v224 (How satisfied with present state of economy), v101 (Feeling about household's income nowadays)

# Models - XGBoost

*XGBoost is an implementation of Gradient Boosted Decision Trees algorithms that offers a high level of efficiency and scalability that can be used for classification.*

- We used a 70/30 train/test split
- Initial run using default parameters scored AUC of 79.22

```
xgb.train(params = params, data = dtrain, nrounds = nrounds,
  watchlist = watchlist, verbose = verbose, print_every_n = print_every_n,
  early_stopping_rounds = early_stopping_rounds, maximize = maximize,
  save_period = save_period, save_name = save_name, xgb_model = xgb_model,
  callbacks = callbacks, eta = 0.1, objective = "binary:logistic",
  max.depth = 5, lambda = 0.9, eval_metric = "auc")
```

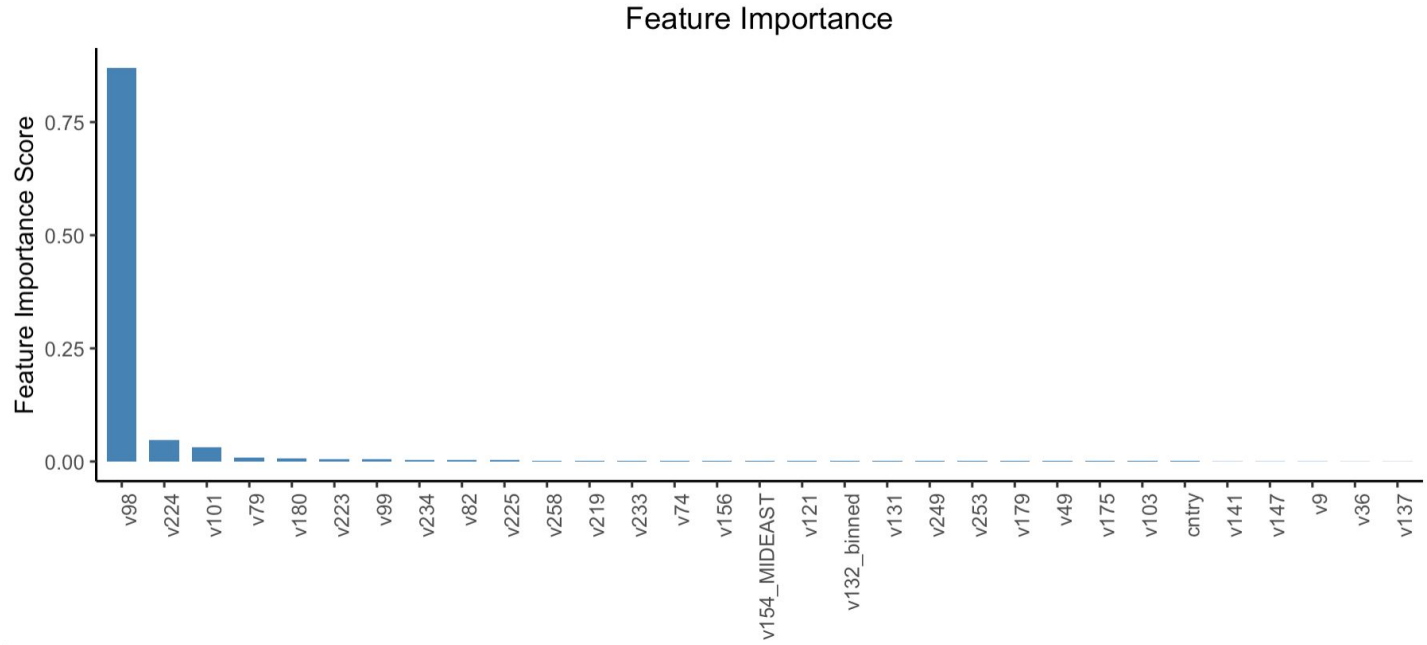
# Models - XGBoost

- Due to efficiency of this algorithm, we tried tuning hyperparameters
- XGBoost has an internal cross validation function that allows us to determine the optimal hyperparameters

- The model returned lowest error at the 3rd iteration (nround / # of boosting iterations)
- Tuned AUC 87.94%
- Submission AUC 86.66%

```
##### xgb.cv 5-folds
 iter train_auc_mean train_auc_std test_auc_mean test_auc_std
    1      0.8810326   0.001503736     0.8743760  0.007627838
    2      0.8910986   0.001342122     0.8783128  0.008273155
    3      0.8979740   0.001722854     0.8794058  0.007833337
    4      0.9032798   0.001793213     0.8778238  0.007532983
    5      0.9078734   0.001808418     0.8769978  0.007380923
    6      0.9116042   0.002028765     0.8754764  0.007902454
    7      0.9153790   0.001999046     0.8735998  0.007884260
    8      0.9189054   0.001777745     0.8730922  0.008349997
    9      0.9215980   0.001577615     0.8717472  0.008528634
   10      0.9248810   0.001494250     0.8702412  0.008061866
```

# Models - XGBoost




- Top 5: v98 (How happy are you), v224 (How satisfied with present economy), v101 (Feeling about household's income), v79 (Felt depressed, how often past week), v80 (Felt everything did as effort, how often past week)

# Models - LightGBM

*LightGBM is a gradient boosting framework that uses tree based learning algorithms.*

- We used a 70/30 train/test split
- Initial run using default parameters scored AUC of 80.95

```
LGBMClassifier(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
               importance_type='split', learning_rate=0.1, max_depth=-1,
               min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0,
               n_estimators=100, n_jobs=-1, num_leaves=31, objective=None,
               random_state=None, reg_alpha=0.0, reg_lambda=0.0, silent=True,
               subsample=1.0, subsample_for_bin=200000, subsample_freq=0)
```

https://lightgbm.readthedocs.io/en/latest/

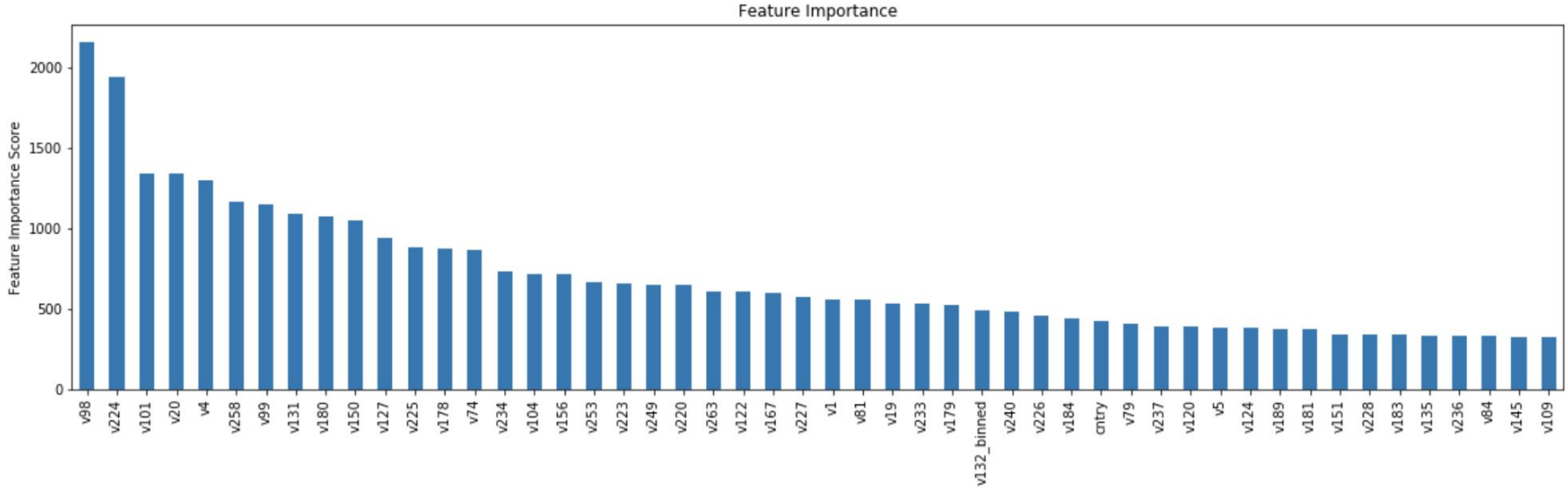
# Models - LightGBM

- Due to efficiency of this algorithm, we tried tuning hyperparameters
- Random Search algorithm with 3-fold CV and 10 iterations for a total of 90 fits
  - Increase in regularization, n_estimators
  - Decrease in learning rate, colsample_bytree

```
{'subsample_for_bin': 200000, 'subsample': 0.5707070707070707, 'reg_lambda': 0.8163265306122448, 'reg_alpha': 0.24489
795918367346, 'objective': 'binary', 'num_leaves': 46, 'n_estimators': 1189, 'min_child_samples': 195, 'metric': 'au
c', 'learning_rate': 0.00635439354601029, 'is_unbalance': False, 'colsample_bytree': 0.7777777777777778, 'boosting_ty
pe': 'gbdt'}
```

- Tuned AUC 81.16%
- Submission AUC 80.63%

# Models - LightGBM




- Top 5: v98 (How happy are you), v224 (How satisfied with present economy), v101 (Feeling about household's income), v20 (Region), v4 (First Ancestry)

# Models - Random Forest

- We used a 70/30 train/test split
- Initial run using default parameters scored AUC of 80.00%

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=3096,
                       verbose=0, warm_start=False)
```

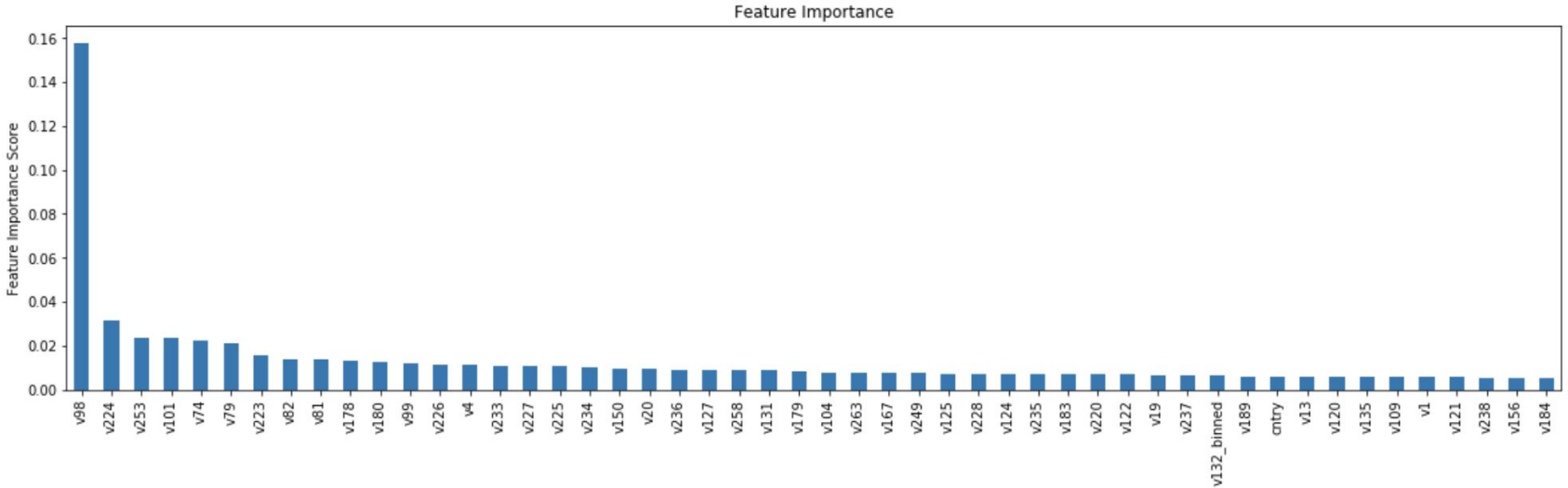
# Models - Random Forest

- Relatively fast to run as well so we tuned hyperparameters for this model
  - Random search algorithm, 3-fold Cross Validation with 10 iterations for total of 90 fits
- Bootstrap, max_depth, increase in n_estimators

```
RandomForestClassifier(bootstrap=False, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=30, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=2, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=1800,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

- Tuned AUC 80.78%
- Submission AUC 88.21%

# Models - Random Forest




- Top 5: v98 (How happy are you), v224 (How satisfied with present economy), v253 (Were happy, how often past week), v101 (Feeling about household's income), v74 (Enjoyed life, how often past week)

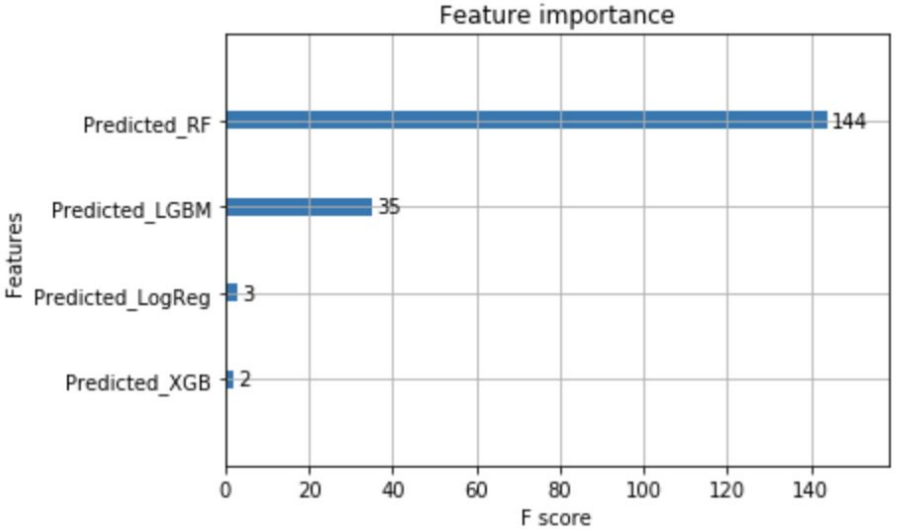
# Final Model

# Final Model - Stacking with XGBoost

- Created new dataset using the predictions from four input models
- Tried HyperOpt but results not significant
- Test AUC 80.70

```
XGBClassifier(base_score=0.5, booster=None, colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints=None,
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints=None,
              n_estimators=100, n_jobs=0, num_parallel_tree=1,
              objective='binary:logistic', random_state=42, reg_alpha=0,
              reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method=None,
              validate_parameters=False, verbosity=None)
```

# Final Model - Stacking with XGBoost




- Stacked model places highest importance on the Random Forest, followed by LightGBM
- Combined train and test sets to train model for submission
- Final AUC **88.67%**

# Conclusion

- Stacking models yields better results when submitted to Kaggle
- Limitation: Unable to run very computationally intensive algorithms on our personal computers
- To further improve model, we would consider revisiting data cleaning/analysis:
    - Applying “smart imputation” technique to more features
    - Narrowing down our features (RFE or other)