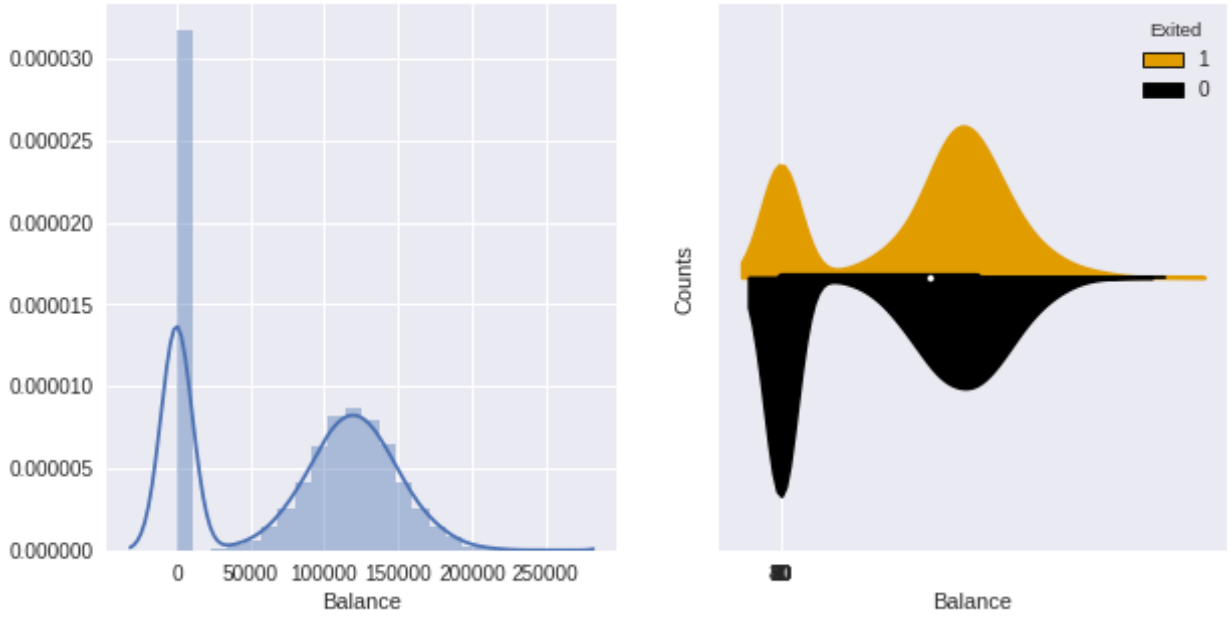
Data Science

Bank Customer Churn Modeling

nhan.cao@beesightsoft.com



Category Balance

Encoder

```
data_encoder['Geography'] = LabelEncoder().fit_transform(data_encoder['Geography'])
data_encoder = data_encoder.join(pd.get_dummies(data_encoder['Gender'], prefix='Gender'))
data_encoder = data_encoder.drop('Gender', axis=1)

data_encoder.loc[ data_encoder['Balance'] <= 118100.59, 'Balance'] = 0
data_encoder.loc[ data_encoder['Balance'] > 118100.59, 'Balance'] = 1
```

Feature selection

DATASET DROP: "RowNumber", "CustomerId", "Surname"

```
Naive Bayes accuracy: 0.770
Logistic Regression accuracy: 0.785
Random Forest accuracy: 0.840000
Linear SVM accuracy: 0.798250
RBF SVM accuracy: 0.798250
K Nearest Neighbor accuracy: 0.736250
ANN accuracy: 0.797500
```

APPROACH 1

Feature Importances

```
Variable: Age                  Importance: 0.32
Variable: NumOfProducts        Importance: 0.18
Variable: EstimatedSalary      Importance: 0.14
Variable: CreditScore          Importance: 0.13
Variable: IsActiveMember       Importance: 0.09
Variable: Tenure               Importance: 0.06
Variable: Geography            Importance: 0.04
Variable: Balance              Importance: 0.02
Variable: HasCrCard            Importance: 0.01
Variable: Gender_Female        Importance: 0.01
Variable: Gender_Male          Importance: 0.01
```

Evaluate

```
Naive Bayes accuracy: 0.767
Logistic Regression accuracy: 0.784
Random Forest accuracy: 0.830750
Linear SVM accuracy: 0.798250
RBF SVM accuracy: 0.798250
K Nearest Neighbor accuracy: 0.736000
ANN accuracy: 0.798750
```

Ranking

	Attributes	Ranking	Support
0	CreditScore	1	True
1	Geography	1	True
2	Age	1	True
3	Tenure	1	True

	Attributes	Ranking	Support
4	**Balance**	**1**	**True**
5	**NumOfProducts**	**1**	**True**
6	**HasCrCard**	**1**	**True**
7	**IsActiveMember**	**1**	**True**
9	**Gender_Female**	**1**	**True**
10	**Gender_Male**	**1**	**True**
8	EstimatedSalary	2	False

Evaluate

```
Naive Bayes accuracy: 0.818
Logistic Regression accuracy: 0.803
Random Forest accuracy: 0.841000
Linear SVM accuracy: 0.795500
RBF SVM accuracy: 0.798250
K Nearest Neighbor accuracy: 0.772750
ANN accuracy: 0.806750
```

APPROACH 2: PCA

Drop: "RowNumber", "CustomerId", "Surname", "HasCrCard", "Gender_Male", "Gender_Female"

Evaluate

```
Naive Bayes accuracy: 0.827
Logistic Regression accuracy: 0.802
Random Forest accuracy: 0.839750
Linear SVM accuracy: 0.853500
RBF SVM accuracy: 0.810500
K Nearest Neighbor accuracy: 0.817500
ANN accuracy: 0.850750
```

APPROACH 3: OUTLIER REMOVAL

Evaluate

```
Naive Bayes accuracy: 0.791
Logistic Regression accuracy: 0.807
Random Forest accuracy: 0.858949
Linear SVM accuracy: 0.811045
RBF SVM accuracy: 0.811045
K Nearest Neighbor accuracy: 0.767132
ANN accuracy: 0.207917
```

APPROACH 4: NEURAL NETWORK APPROACH

Using PCA data

```
# 1st layer: 23 nodes, input shape[1] nodes, RELU
model.add(Dense(23, input_dim=X_train.shape[1], kernel_initializer='uniform', activation='relu'))
# 2nd layer: 17 nodes, RELU
model.add(Dense(17, kernel_initializer='uniform', activation = 'relu'))
# 3nd layer: 15 nodes, RELU
model.add(Dense(15, kernel_initializer='uniform', activation='relu'))
# 4nd layer: 11 nodes, RELU
model.add(Dense(11, kernel_initializer='uniform', activation='relu'))
# 5nd layer: 9 nodes, RELU
model.add(Dense(9, kernel_initializer='uniform', activation='relu'))
# 6nd layer: 7 nodes, RELU
model.add(Dense(7, kernel_initializer='uniform', activation='relu'))
# 7nd layer: 5 nodes, RELU
model.add(Dense(5, kernel_initializer='uniform', activation='relu'))
# 8nd layer: 2 nodes, RELU
model.add(Dense(2, kernel_initializer='uniform', activation='relu'))
# output layer: dim=1, activation sigmoid
model.add(Dense(1, kernel_initializer='uniform', activation='sigmoid' ))
# Compile the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

NB_EPOCHS = 100
BATCH_SIZE = 23
```

Evaluate:

```
Epoch 00100: val_acc did not improve from 0.85275
```

APPROACH 5: BAGGING BOOSTING AND STACKING

Using PCA data

5-fold cross validation

```
Accuracy: 0.82 (+/- 0.00) [NaiveBayesLearning]
Accuracy: 0.81 (+/- 0.00) [LogisticRegressionLearning]
Accuracy: 0.84 (+/- 0.00) [RandomForestLearning]
Accuracy: 0.86 (+/- 0.00) [SVMLearningLinear]
Accuracy: 0.82 (+/- 0.00) [SVMLearningRBF]
Accuracy: 0.83 (+/- 0.01) [KNNLearning]
Accuracy: 0.85 (+/- 0.01) [ANNLearning]
```

EnsembleVoteClassifier

```
Accuracy: 0.84 (+/- 0.01) [RandomForestLearning]
Accuracy: 0.86 (+/- 0.00) [SVMLearningLinear]
```

```
Accuracy: 0.84 (+/- 0.02) [ANNModel]
Accuracy: 0.86 (+/- 0.01) [Ensemble]
```

MajorityVoteClassifier

```
ROC AUC: 0.84 (+/- 0.01) [RandomForestLearning]
ROC AUC: 0.86 (+/- 0.00) [SVMLearningLinear]
ROC AUC: 0.84 (+/- 0.02) [ANN]
ROC AUC: 0.85 (+/- 0.00) [Majority voting]
```

BaggingClassifier: using RFModel

```
Decision tree train/test accuracies 1.000/0.788
Bagging train/test accuracies 0.957/0.874
```

AdaBoostClassifier

```
Decision tree train/test accuracies 1.000/0.788
AdaBoost train/test accuracies 1.000/0.789
```

StackingClassifier

```
Accuracy: 0.84 (+/- 0.01) [RandomForestLearning]
Accuracy: 0.86 (+/- 0.00) [SVMLearningLinear]
Accuracy: 0.84 (+/- 0.02) [ANN]
Accuracy: 0.84 (+/- 0.01) [Stacking Classifier]
```

Summaries

Using Bagging on RandomForest can make up to 87.4%