Will the customers we contact accept a Credit Card offer?

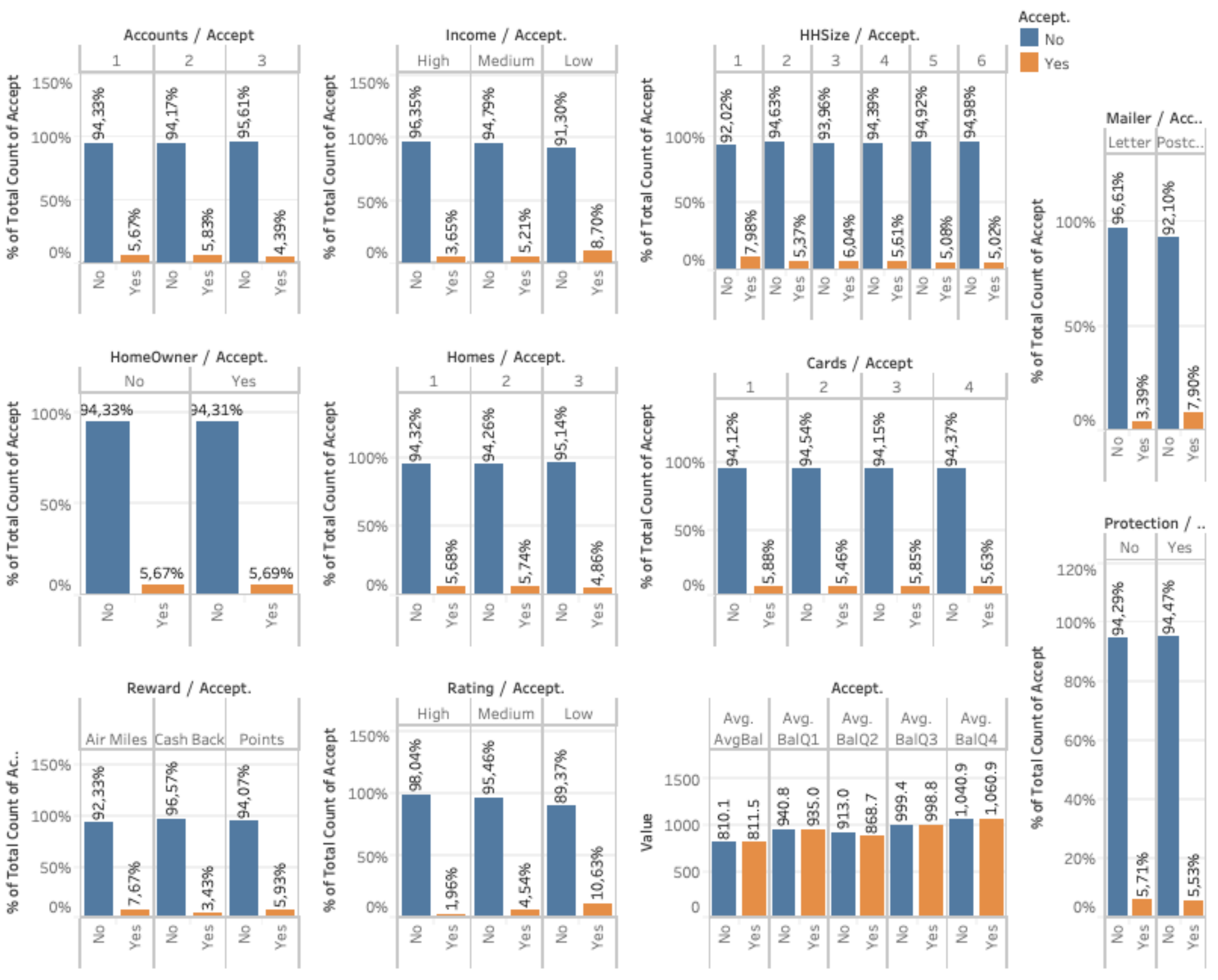
CLASSIFICATION MODEL

WEEK 5 PROJECT RUNTHROUGH

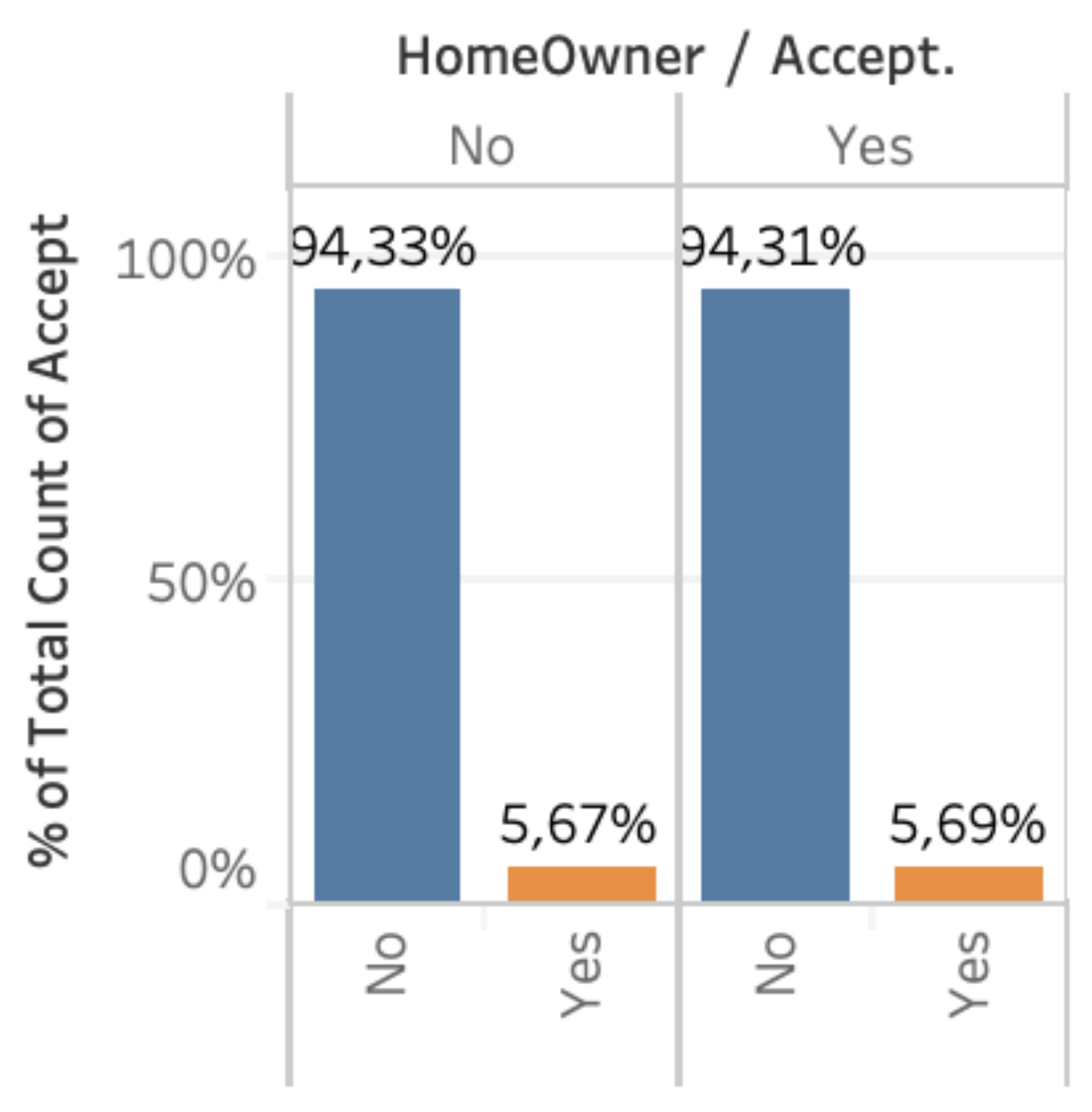
David Magraner Villalba

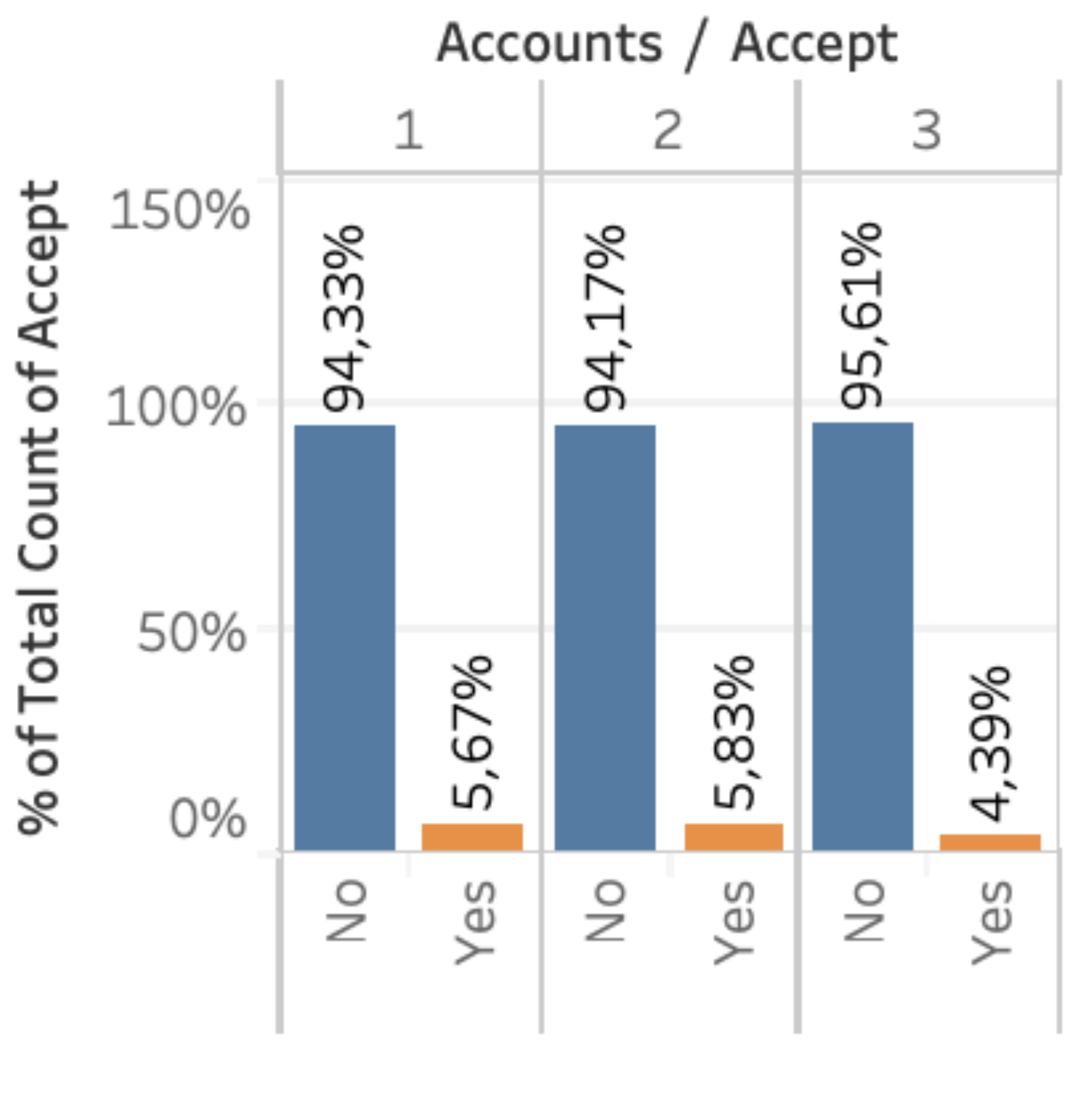
IRONHACK BERLIN

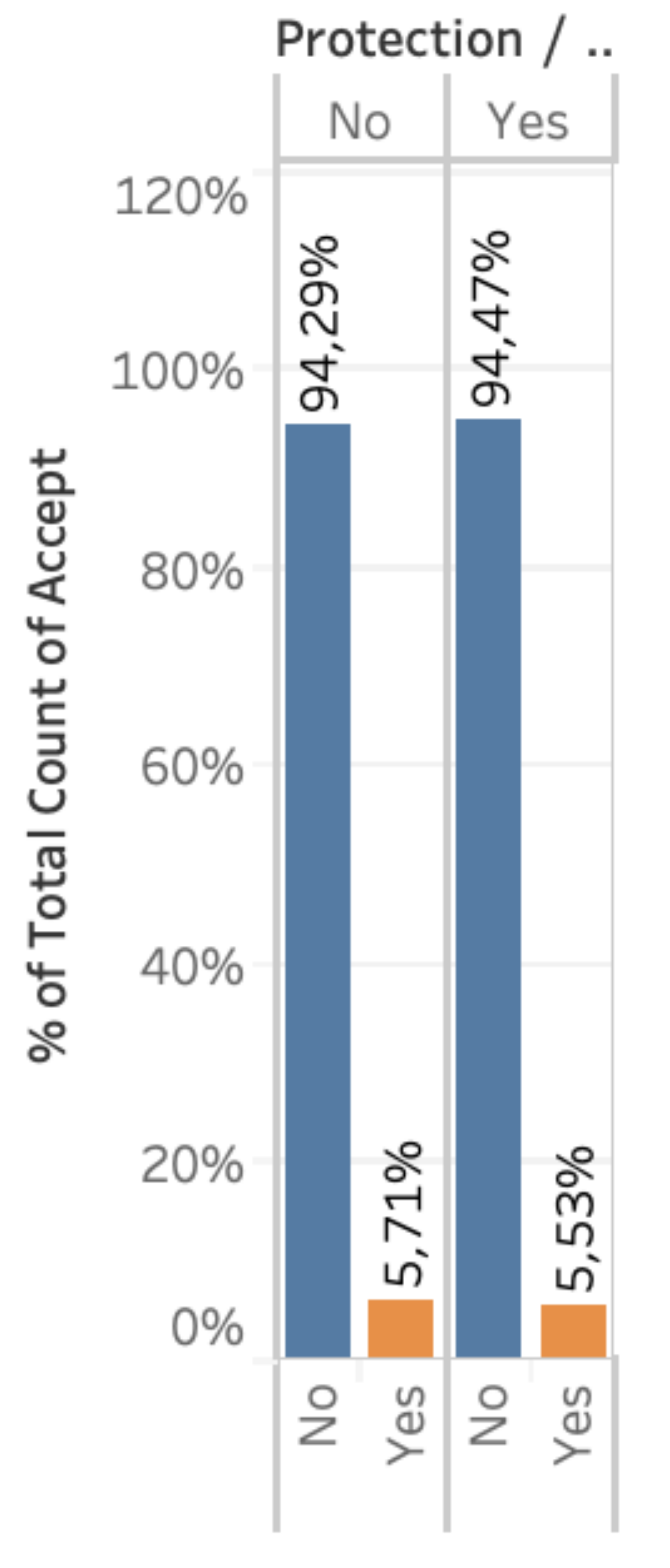
FAST FIRST DATA VIZ

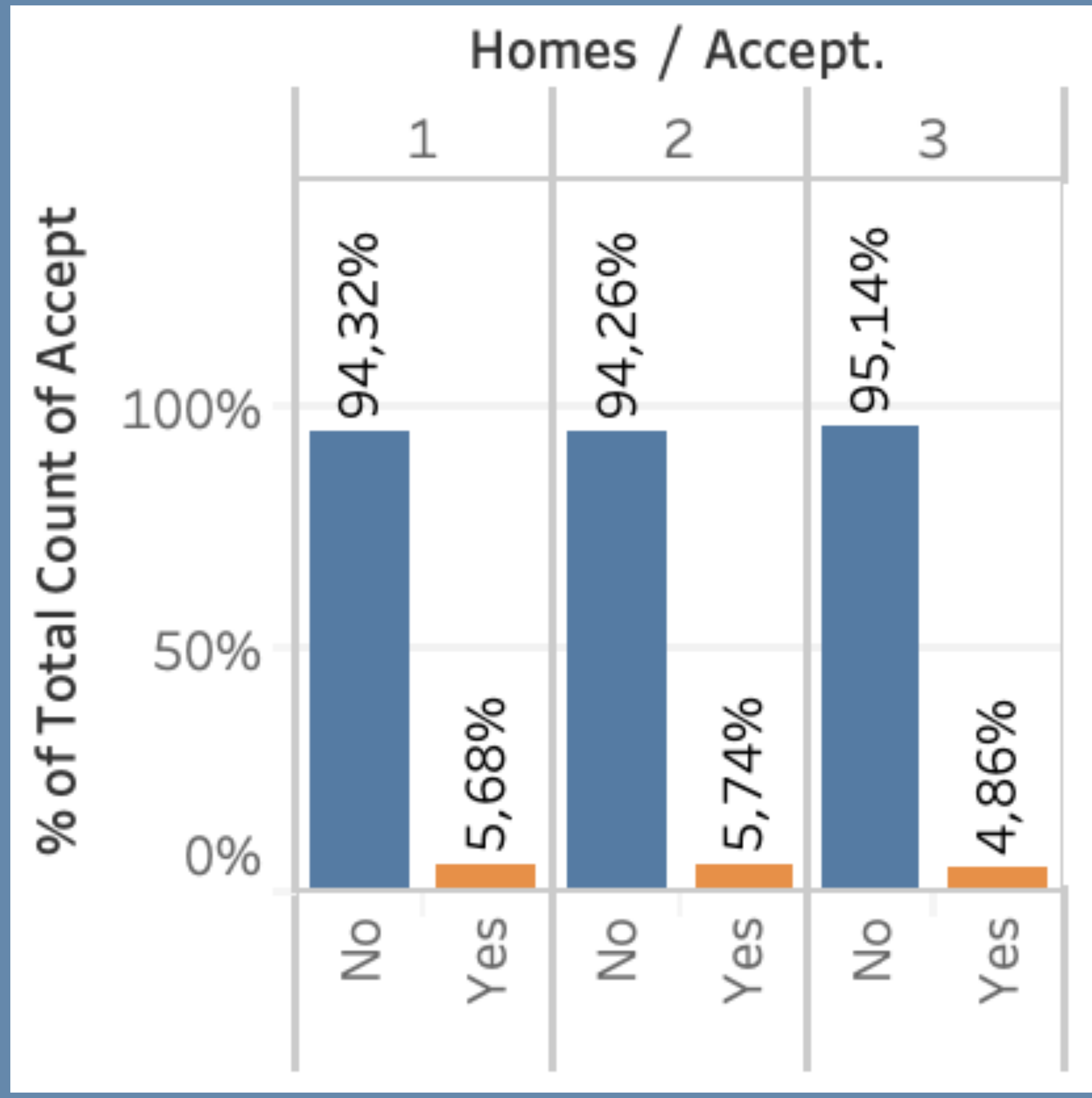


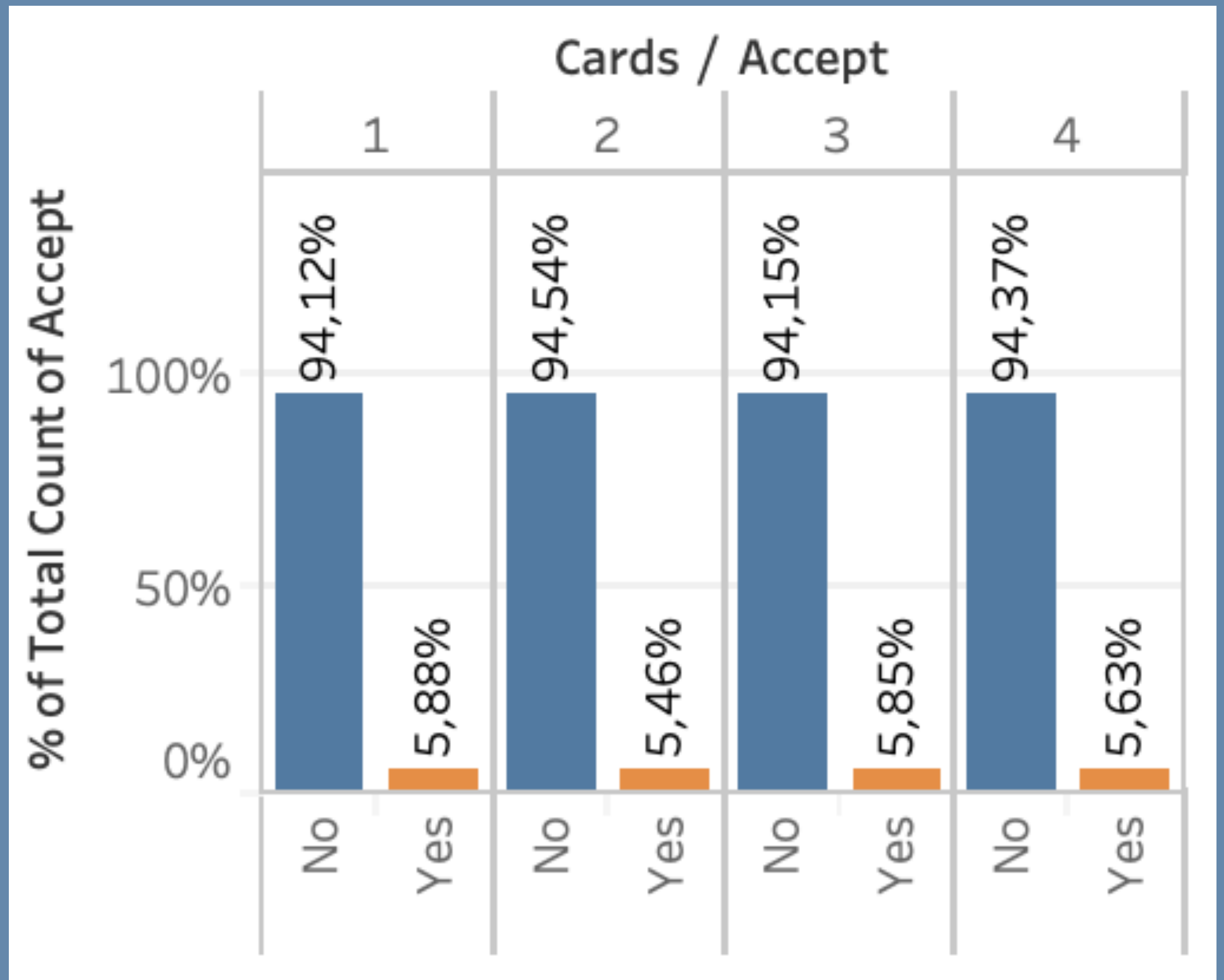
SEEMINGLY NOT SIGNIFICANT TRAITS



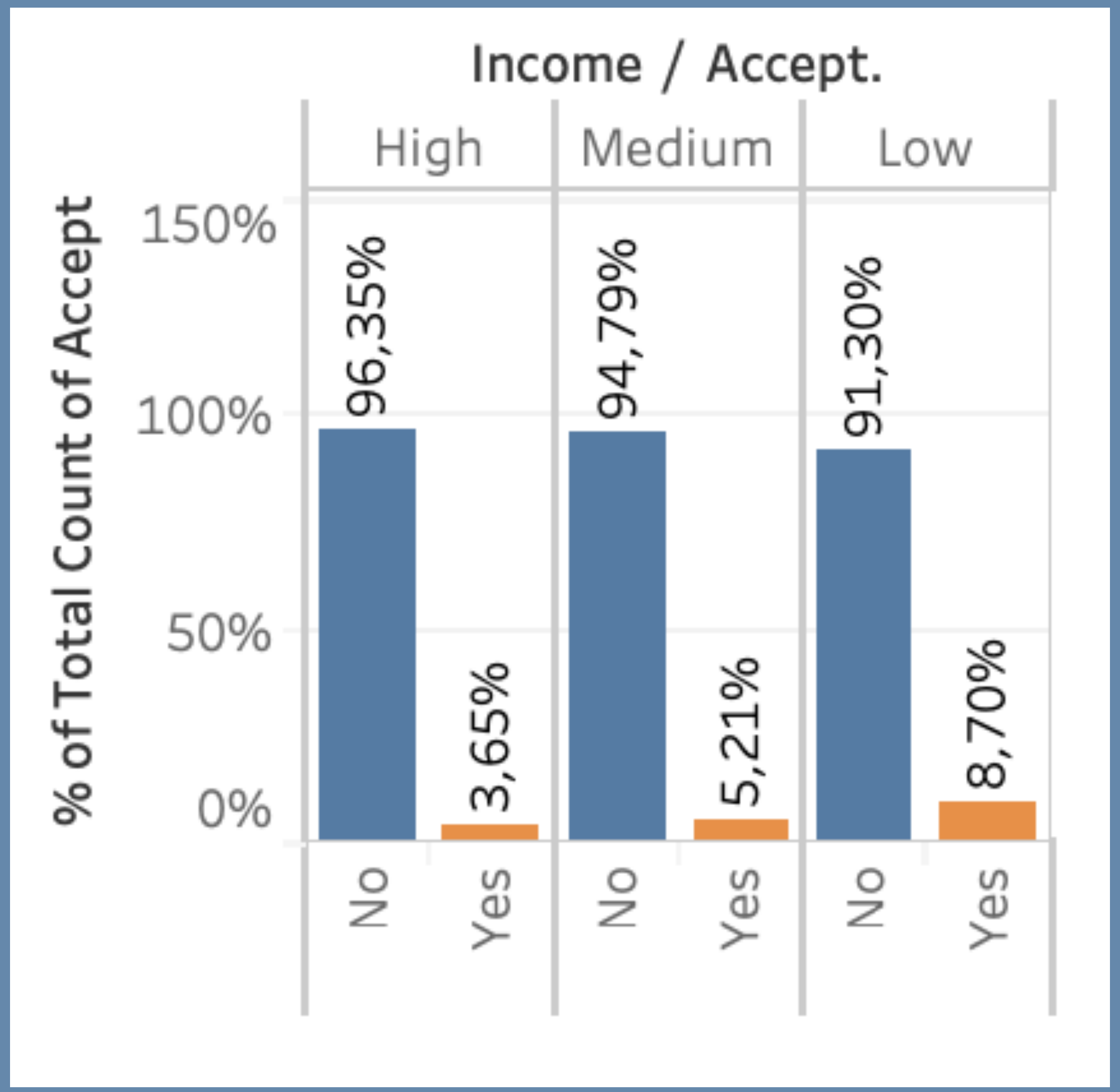


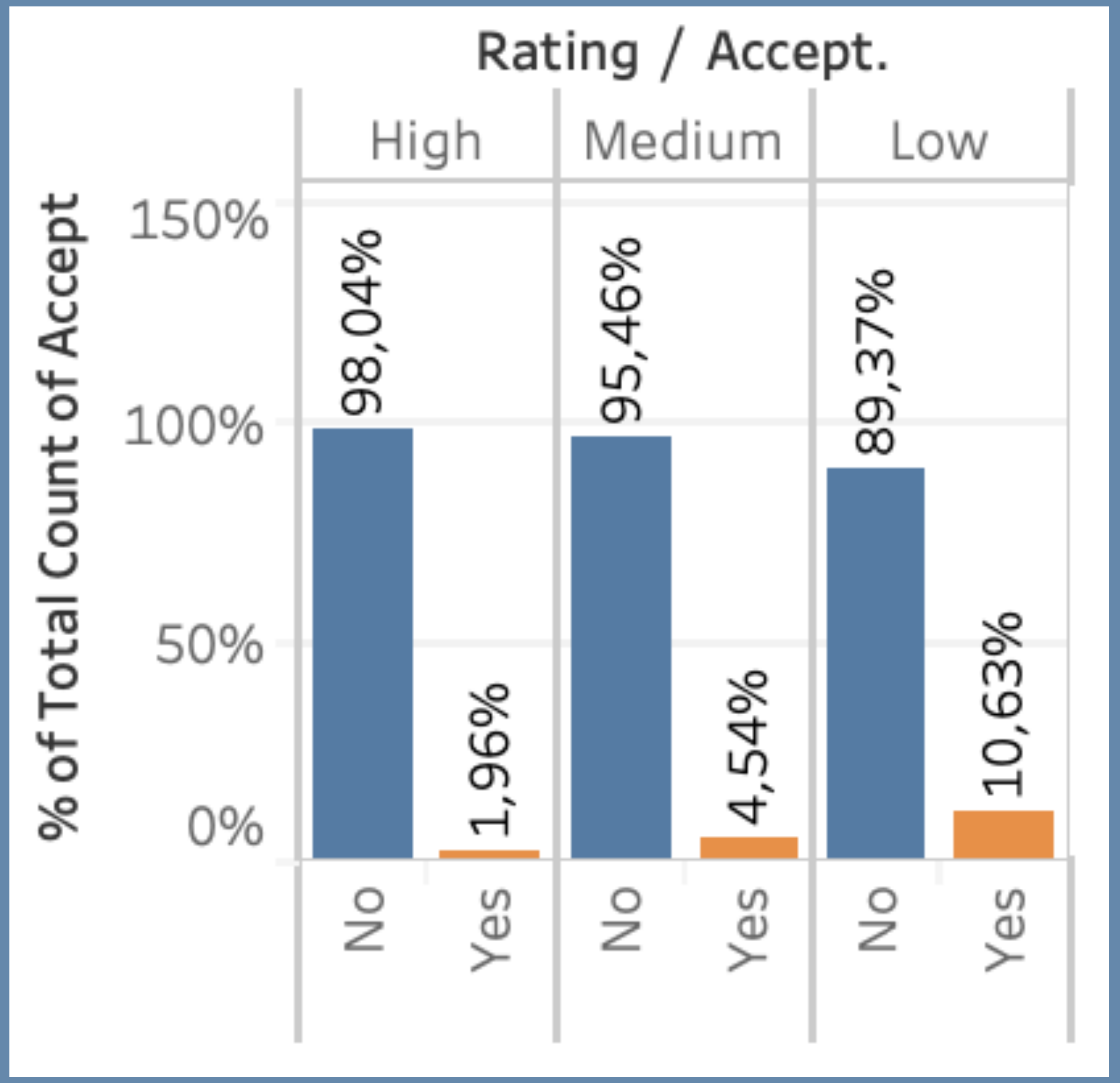


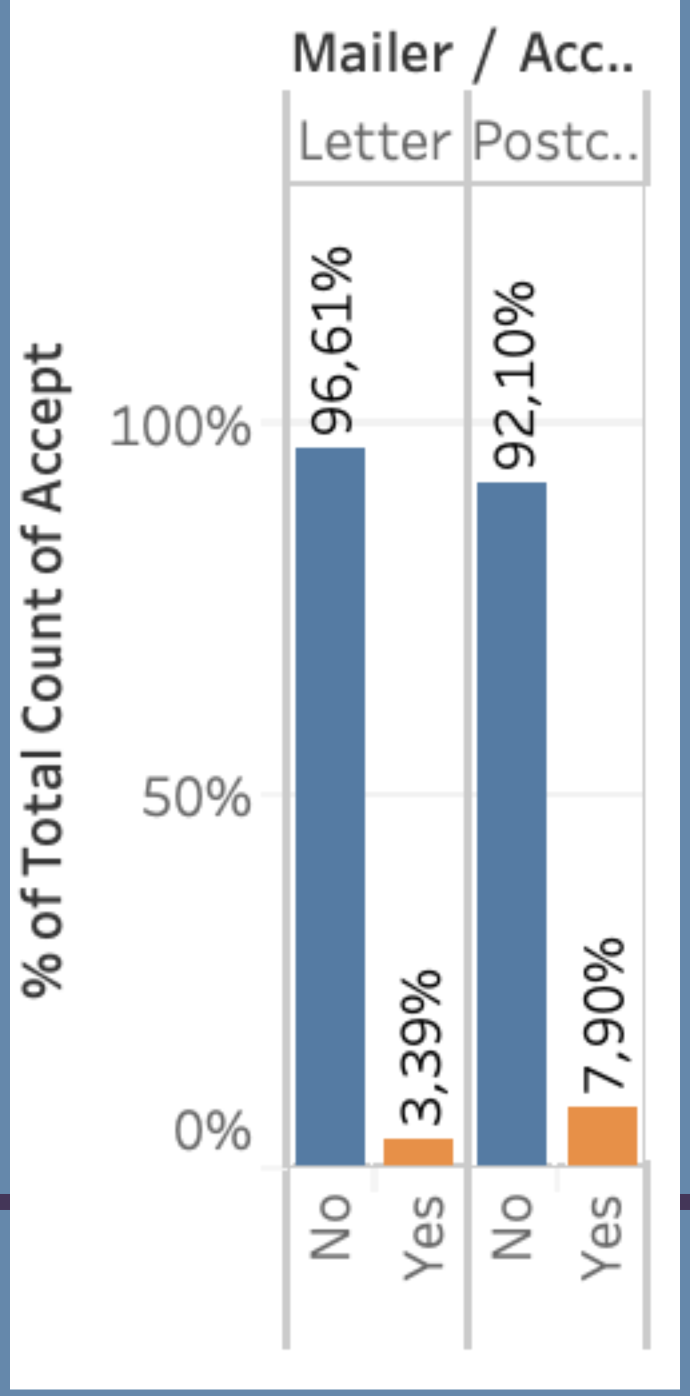


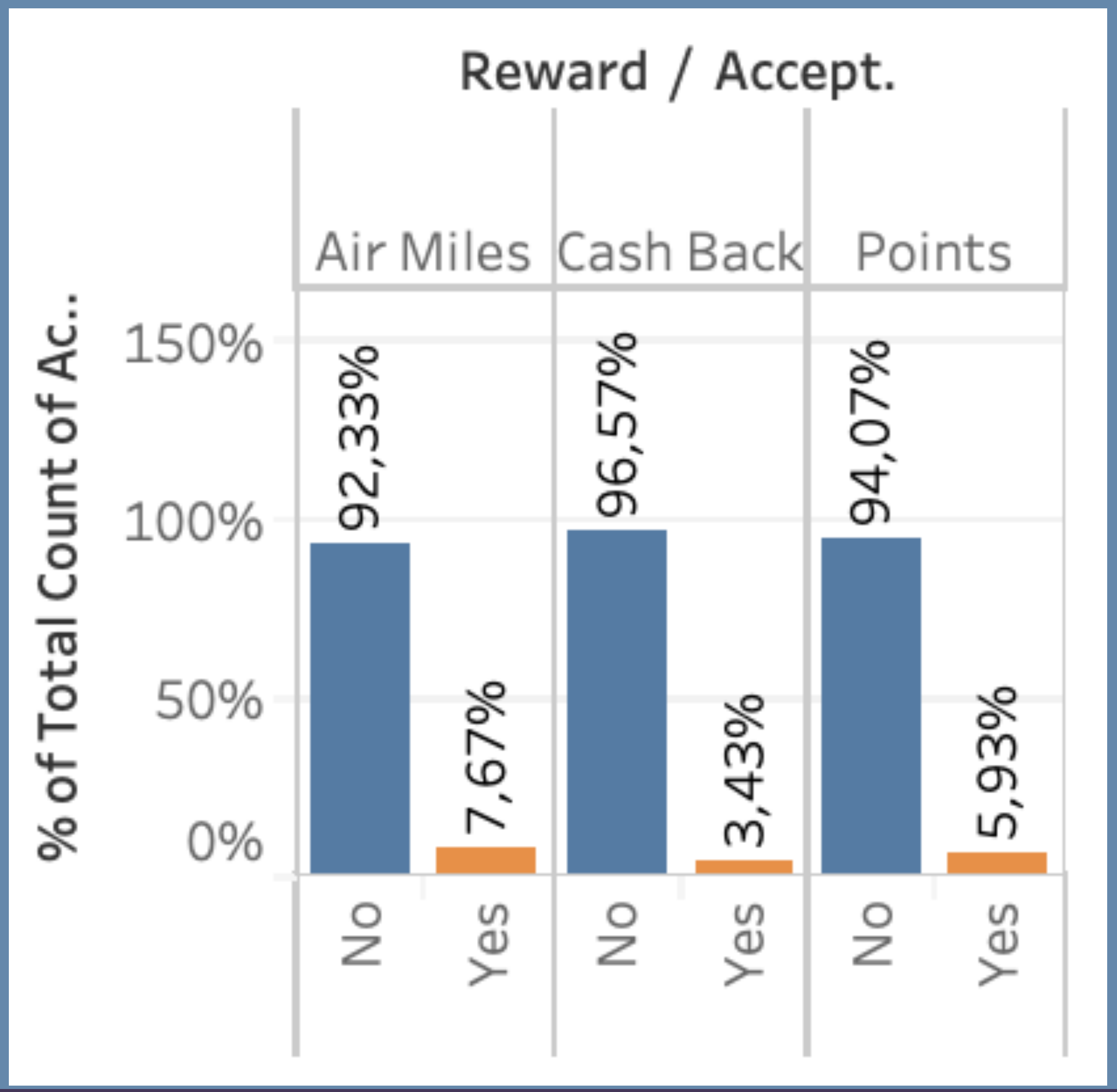


POTENTIALLY SIGNIFICANT TRAITS



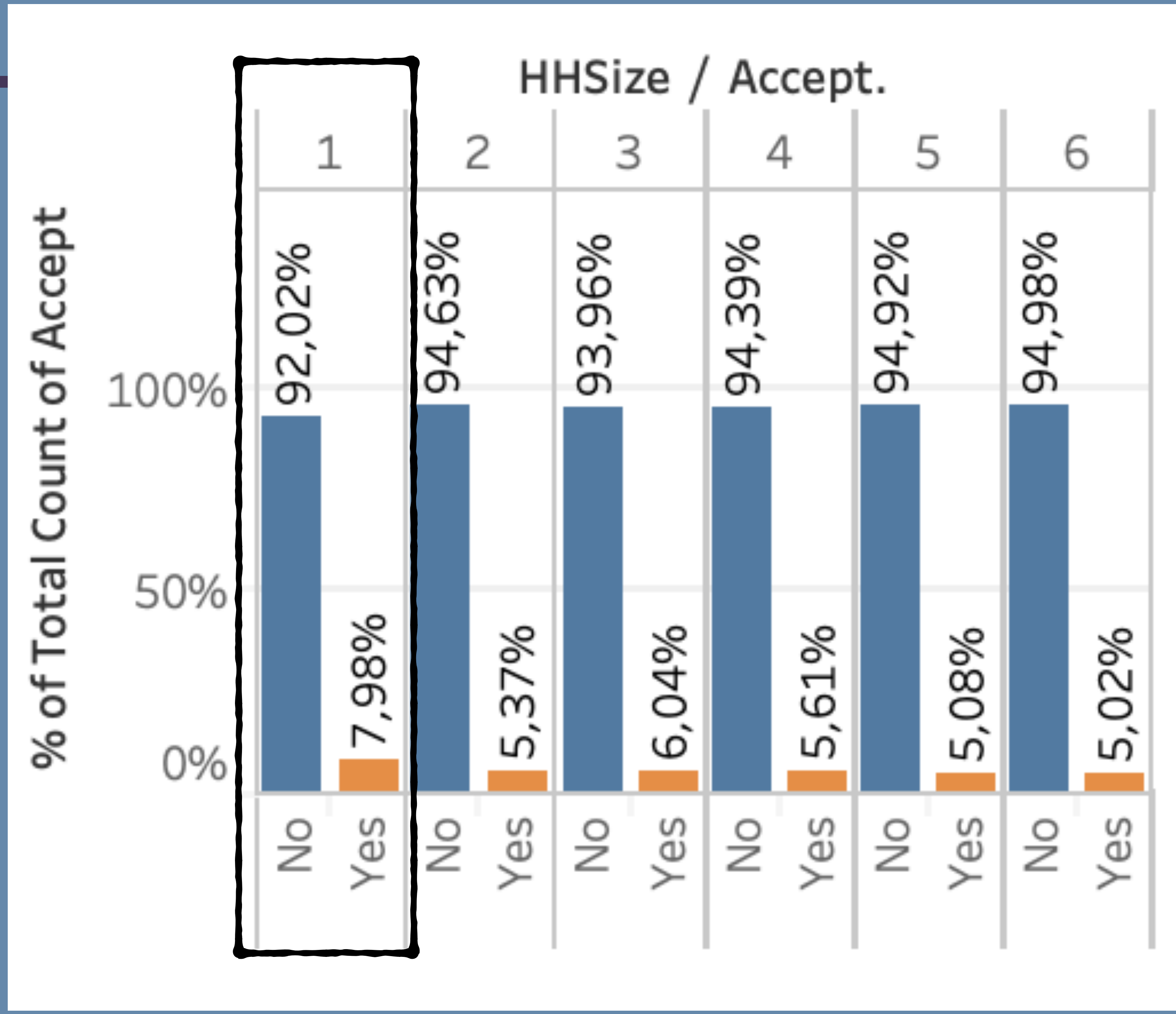




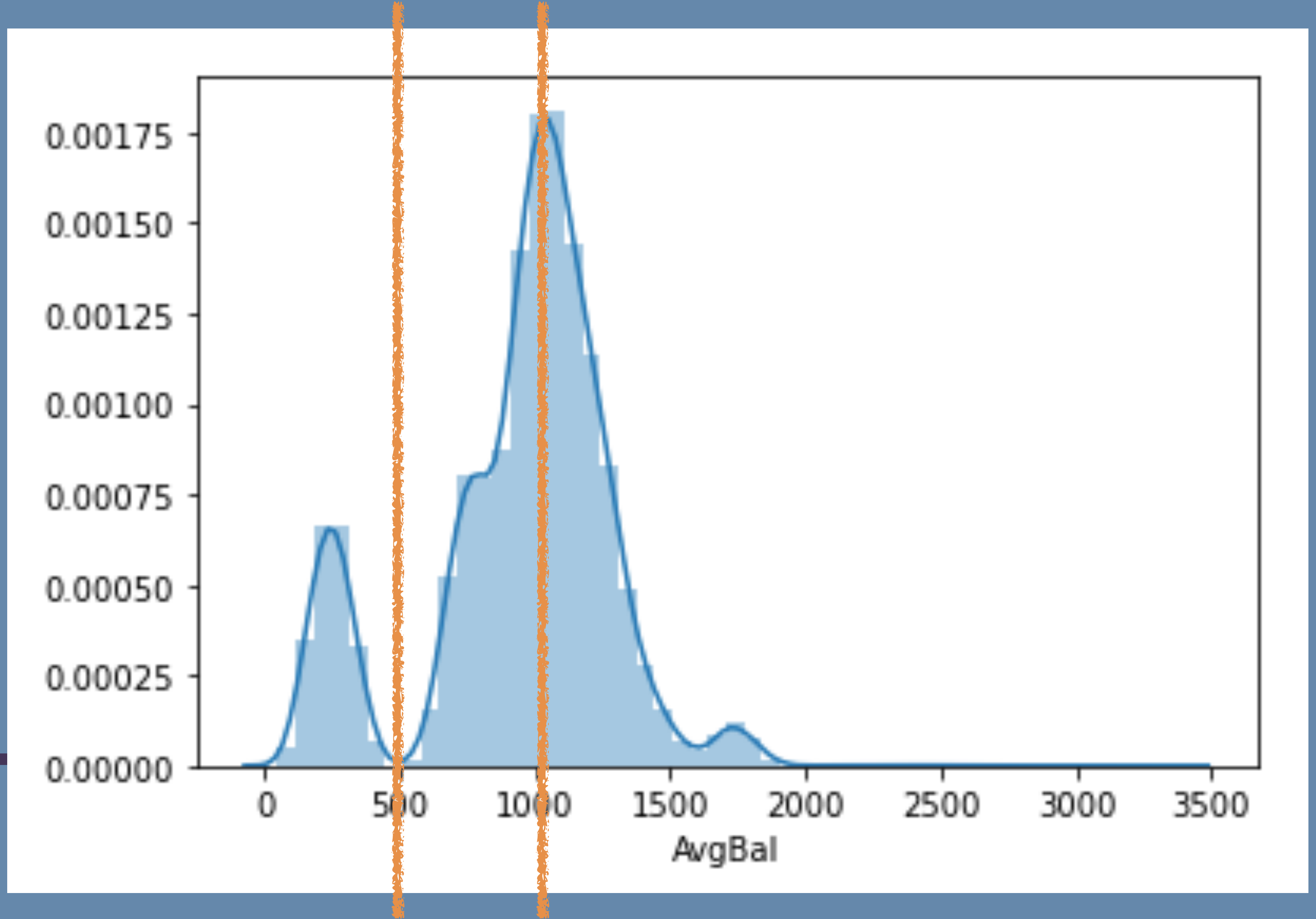


OTHER TRAITS THAT COULD BE IMPORTANT

```
data['LiveAlone'] = np.where(data['HHSize'] == 1, 1, 0)
```




```
data['LowBalance'] = np.where(data['AvgBal'] <= 500, 1, 0)
data['HighBalance'] = np.where(data['AvgBal'] >= 1000, 1, 0)
```



DATASET

Accept	AirMiles	Points	Letter	Income	Rating	LiveAlone	LowBalance	AvgBal
0	0	1	0	1	3	0	0	1175
0	0	1	1	2	1	0	0	811
0	0	0	1	3	3	0	0	1754
0	1	0	0	3	3	0	0	689
0	0	0	1	1	1	0	0	1018
0	0	0	0	2	3	0	0	1130
0	1	0	0	1	1	1	0	877
1	1	0	1	2	1	0	0	769
0	0	0	0	2	1	0	0	709
0	0	1	0	3	2	1	0	690
0	0	1	1	2	3	0	0	863
0	0	0	0	1	1	0	0	733
0	0	1	0	1	3	0	0	771
0	1	0	0	1	2	0	0	659
0	1	0	1	2	1	0	0	1258

MAIN ISSUE:
DATA IMBALANCE

5.68%	94.32%
YES	NO

NORMAL MODEL WILL BE BIASED TOWARDS PREDICTING NO FOR ALL OBSERVATIONS

TREATING DATA IMBALANCE

NORMAL MODEL WILL BE BIASED TOWARDS PREDICTING NO FOR ALL OBSERVATIONS

OUR JOB IS TO MAKE A MODEL THAT FOCUSES ON PREDICTING YESES AS ACCURATELY AS POSSIBLE.

EVALUATING IMBALANCED MODEL

OUR MODEL COULD BE IMBALANCED: PREDICTS NOS BY DEFAULT. WE WANT TO KNOW HOW OUR MODEL REDUCES FN AND ENSURE YES IS PREDICTED AS YES.

SENSITIVITY: TP/(TP+FN)

1.A LOGISTIC MODEL WITH COMPLETE DATASET

```python
# This is the model for the complete dataset, df1.

for sam in [1, 0.9, 0.8, 0.66, 0.5, 0.4, 0.33, 0.25, 0.2]:
    for test in [0.2, 0.25, 0.3]:
        undersample = RandomUnderSampler(sampling_strategy=sam)
        X_und, y_und = undersample.fit_resample(X1, y1)
        X_train, X_test, y_train, y_test = train_test_split(X_und, y_und, test_size=test, random_state=42)
        classifier = LogisticRegression(solver='liblinear', class_weight='balanced')
        classifier.fit(X_train, y_train)
        y_pred = classifier.predict(X_test)
        pre = classification_report(y_test, y_pred, output_dict=True)['1']['precision']*100
        acc = classification_report(y_test, y_pred, output_dict=True)['1']['recall']*100
        print(f'Log model with a {sam:.2f} ratio, test size of {test:.2f}: {acc:.2f}% of sensitivity and a
```

```
Log model with a 1.00 ratio, test size of 0.20: 73.50% of sensitivity and a 63.64% of precision.
Log model with a 1.00 ratio, test size of 0.25: 75.21% of sensitivity and a 67.66% of precision.
Log model with a 1.00 ratio, test size of 0.30: 75.51% of sensitivity and a 70.70% of precision.
Log model with a 0.90 ratio, test size of 0.20: 64.79% of sensitivity and a 66.99% of precision.
Log model with a 0.90 ratio, test size of 0.25: 66.54% of sensitivity and a 64.29% of precision.
Log model with a 0.90 ratio, test size of 0.30: 67.54% of sensitivity and a 67.54% of precision.
```

1.B LOGISTIC MODEL WITH REDUCED DATASET

```python
# This is the model for the reduced dataset, df2.

for sam in [1, 0.9, 0.8, 0.66, 0.5, 0.4, 0.33, 0.25, 0.2]:
    for test in [0.2, 0.25, 0.3]:
        undersample = RandomUnderSampler(sampling_strategy=sam)
        X_und, y_und = undersample.fit_resample(X2, y2)
        X_train, X_test, y_train, y_test = train_test_split(X_und, y_und, test_size=test, random_state=42)
        classifier = LogisticRegression(solver='liblinear', class_weight='balanced')
        classifier.fit(X_train, y_train)
        y_pred = classifier.predict(X_test)
        pre = classification_report(y_test, y_pred, output_dict=True)['1']['precision']*100
        acc = classification_report(y_test, y_pred, output_dict=True)['1']['recall']*100
        print(f'Log model with a {sam:.2f} ratio, test size of {test:.2f}: {acc:.2f}% of sensitivity and a
```

```
Log model with a 1.00 ratio, test size of 0.20: 74.50% of sensitivity and a 69.63% of precision.
Log model with a 1.00 ratio, test size of 0.25: 81.40% of sensitivity and a 68.17% of precision.
Log model with a 1.00 ratio, test size of 0.30: 76.19% of sensitivity and a 66.87% of precision.
Log model with a 0.90 ratio, test size of 0.20: 66.67% of sensitivity and a 68.27% of precision.
Log model with a 0.90 ratio, test size of 0.25: 67.32% of sensitivity and a 64.07% of precision.
Log model with a 0.90 ratio, test size of 0.30: 71.15% of sensitivity and a 63.82% of precision.
Log model with a 0.80 ratio, test size of 0.20: 74.26% of sensitivity and a 64.38% of precision.
```

2. K-NEAREST NEIGHBOURS MODEL

```
# This is the model for the reduced dataset, df2.

for sam in [1, 0.9, 0.8]:
    for test in [0.2, 0.25, 0.3]:
        for k in [3, 5, 7, 9]:
            undersample = RandomUnderSampler(sampling_strategy=sam)
            X_und, y_und = undersample.fit_resample(X2, y2)
            X_train, X_test, y_train, y_test = train_test_split(X_und, y_und, test_size=test, random_state=42)
            nbrs = NearestNeighbors(n_neighbors=k)
            nbrs.fit(X_train, y_train)
            y_pred = classifier.predict(X_test)
            pre = classification_report(y_test, y_pred, output_dict=True)['1']['precision']*100
            acc = classification_report(y_test, y_pred, output_dict=True)['1']['recall']*100
            print(f'Log model with a {sam:.2f} ratio, test size of {test:.2f}, K = {k:.0f}: {acc:.2f}% of sens
```

```
Log model with a 1.00 ratio, test size of 0.20, K = 3: 72.50% of sensitivity and a 68.08% of precision.
Log model with a 1.00 ratio, test size of 0.20, K = 5: 72.50% of sensitivity and a 68.72% of precision.
Log model with a 1.00 ratio, test size of 0.20, K = 7: 72.50% of sensitivity and a 68.72% of precision.
Log model with a 1.00 ratio, test size of 0.20, K = 9: 72.50% of sensitivity and a 70.73% of precision.
```

3. SUPPORT VECTOR MACHINE (SVM)

```python
# This is the model for the reduced dataset, df2.

for sam in [1, 0.9, 0.8]:
    for test in [0.2, 0.25, 0.3]:
        undersample = RandomUnderSampler(sampling_strategy=sam)
        X_und, y_und = undersample.fit_resample(X2, y2)
        X_train, X_test, y_train, y_test = train_test_split(X_und, y_und, test_size=test, random_state=42)
        clas = svm.SVC()
        clas.fit(X_train, y_train)
        y_pred = classifier.predict(X_test)
        pre = classification_report(y_test, y_pred, output_dict=True)['1']['precision']*100
        acc = classification_report(y_test, y_pred, output_dict=True)['1']['recall']*100
        print(f'Log model with a {sam:.2f} ratio, test size of {test:.2f}, K = {k:.0f}: {acc:.2f}% of sens
```

```
Log model with a 1.00 ratio, test size of 0.20, K = 9: 72.50% of sensitivity and a 65.91% of precision.
Log model with a 1.00 ratio, test size of 0.25, K = 9: 73.55% of sensitivity and a 70.08% of precision.
Log model with a 1.00 ratio, test size of 0.30, K = 9: 74.49% of sensitivity and a 68.87% of precision.
Log model with a 0.90 ratio, test size of 0.20, K = 9: 64.79% of sensitivity and a 68.32% of precision.
```

4. DECISION TREE MODEL

```
# This is the model for the reduced dataset, df2.

for sam in [1, 0.9, 0.8]:
    for test in [0.2, 0.25, 0.3]:
        undersample = RandomUnderSampler(sampling_strategy=sam)
        X_und, y_und = undersample.fit_resample(X2, y2)
        X_train, X_test, y_train, y_test = train_test_split(X_und, y_und, test_size=test, random_state=42)
        clas = tree.DecisionTreeClassifier()
        clas.fit(X_train, y_train)
        y_pred = classifier.predict(X_test)
        pre = classification_report(y_test, y_pred, output_dict=True)['1']['precision']*100
        acc = classification_report(y_test, y_pred, output_dict=True)['1']['recall']*100
        print(f'Log model with a {sam:.2f} ratio, test size of {test:.2f}, K = {k:.0f}: {acc:.2f}% of sensi
```

```
Log model with a 1.00 ratio, test size of 0.20, K = 9: 72.50% of sensitivity and a 73.98% of precision.
Log model with a 1.00 ratio, test size of 0.25, K = 9: 73.55% of sensitivity and a 66.92% of precision.
Log model with a 1.00 ratio, test size of 0.30, K = 9: 74.49% of sensitivity and a 67.18% of precision.
Log model with a 0.90 ratio, test size of 0.20, K = 9: 64.79% of sensitivity and a 67.65% of precision.
```

5. RANDOM FOREST MODEL

```
# This is the model for the reduced dataset, df2.

for sam in [1, 0.9, 0.8]:
    for test in [0.2, 0.25, 0.3]:
        undersample = RandomUnderSampler(sampling_strategy=sam)
        X_und, y_und = undersample.fit_resample(X2, y2)
        X_train, X_test, y_train, y_test = train_test_split(X_und, y_und, test_size=test, random_state=42)
        clasRF = RandomForestClassifier(max_depth=5, random_state=42, n_estimators = 100)
        clasRF.fit(X_train, y_train)
        y_pred = classifier.predict(X_test)
        pre = classification_report(y_test, y_pred, output_dict=True)['1']['precision']*100
        acc = classification_report(y_test, y_pred, output_dict=True)['1']['recall']*100
        print(f'Log model with a {sam:.2f} ratio, test size of {test:.2f}, K = {k:.0f}: {acc:.2f}% of sensi
```

```
Log model with a 1.00 ratio, test size of 0.20, K = 9: 72.50% of sensitivity and a 69.05% of precision.
Log model with a 1.00 ratio, test size of 0.25, K = 9: 73.55% of sensitivity and a 69.80% of precision.
Log model with a 1.00 ratio, test size of 0.30, K = 9: 74.49% of sensitivity and a 72.04% of precision.
Log model with a 0.90 ratio, test size of 0.20, K = 9: 64.79% of sensitivity and a 62.73% of precision.
```

CHOSEN MODEL STATISTICS

```python
undersample = RandomUnderSampler(sampling_strategy=1.0)
X_und, y_und = undersample.fit_resample(X2, y2)
X_train, X_test, y_train, y_test = train_test_split(X_und, y_und, test_size=0.25, random_state=42)
classifier = LogisticRegression(solver='liblinear', class_weight='balanced')
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
print(classification_report(y_test, y_pred, output_dict=False))
```

```
              precision    recall  f1-score   support

           0       0.79      0.64      0.70       269
           1       0.67      0.81      0.73       242

    accuracy                           0.72       511
   macro avg       0.73      0.72      0.72       511
weighted avg       0.73      0.72      0.72       511
```

OTHER IMPORTANT ASPECTS TO NOTE

'AVGBAL', 'HIGHBALANCE', 'LOWBALANCE' COULD POTENTIALLY LEAD TO MULTICOLLINEARITY (SORT OF SAME INFO). I DROPPED 'AVGBAL' AND RERUN THE MODELS, AND FOUND NO STATISTICALLY SIGNIFICANT DIFFERENCES BUT A SLIGHT REDUCTION IN SENSITIVITY.