Credit Risk Analysis

Overview:
In this project, we are going to use techniques to build, train, and evaluate models with unbalanced classes using imbalanced-learn and scikit-learn libraries. We are going to use the credit card credit dataset from LendingClub, a peer-to-peer lending services company to:

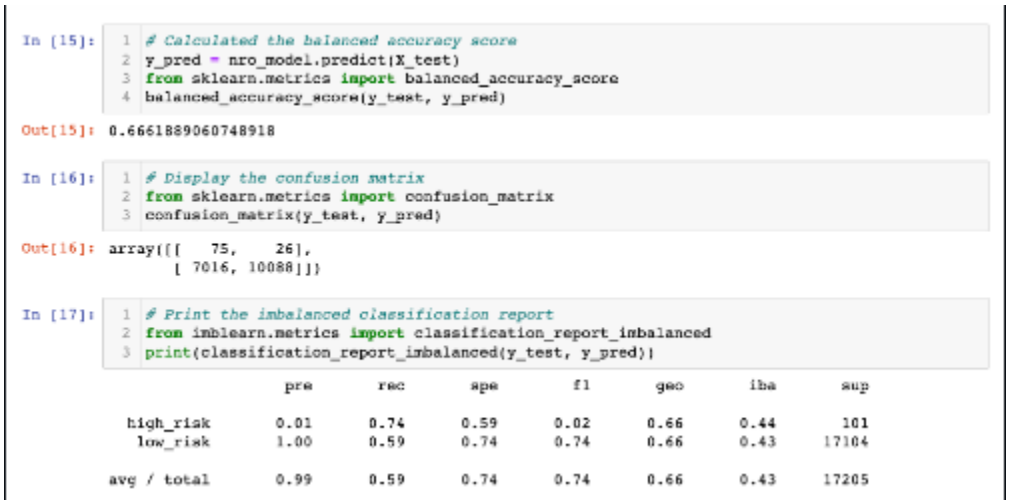
1) Oversample the data using the RandomOverSampler algorithms.
2) Oversample the data using the SMOTE algorithms.
3) Undersample the data using the ClusterCentroids algorithm.
4) Use a combinatorial approach of oversampling and undersampling using the SMOTEENN algorithm.
5) Compare two new machine learning models that reduce bias, BalancedRandomForestClassifier and EasyEnsembleClassifier, to predict credit risk.

Objective:
The aim of this project is to evaluate the performance of 6 machine learning models in order to find the most suitable one to predict credit risk.

Resources:
Data Source: LoanStats_2019Q1.csv
Software: Python, Jupyter Notebook
Scripts: credit_risk_resampling.ipynb, credit_risk_ensemble.ipynb

Analysis of Data:
First Model: Naive Random Oversampling Model

Naive Random Oversampling Evaluation:



In [15]:
```
# Calculated the balanced accuracy score
y_pred = nro_model.predict(X_test)
from sklearn.metrics import balanced_accuracy_score
balanced_accuracy_score(y_test, y_pred)
```

Out[15]: 0.6661889060748918

In [16]:
```
# Display the confusion matrix
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

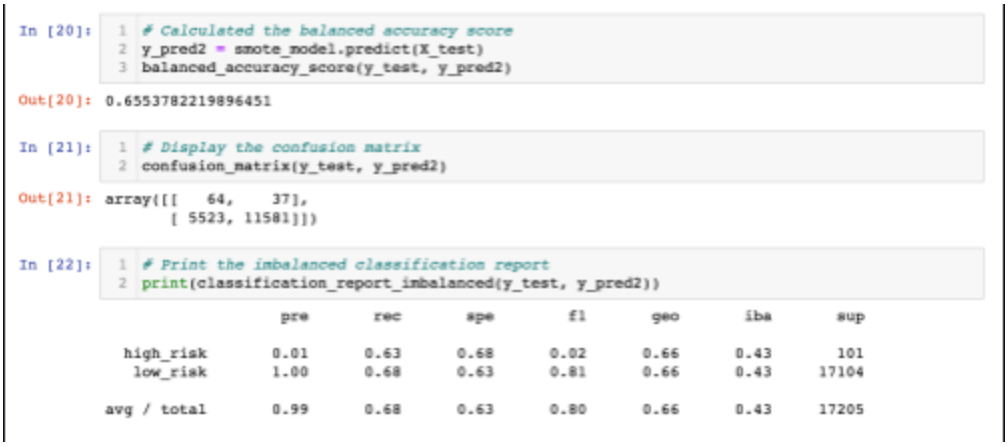
Out[16]:
```
array([[   75,    26],
       [ 7016, 10088]])
```

In [17]:
```
# Print the imbalanced classification report
from imblearn.metrics import classification_report_imbalanced
print(classification_report_imbalanced(y_test, y_pred))
```

	pre	rec	spe	f1	geo	iba	sup
high_risk	0.01	0.74	0.59	0.02	0.66	0.44	101
low_risk	1.00	0.59	0.74	0.74	0.66	0.43	17104
avg / total	0.99	0.59	0.74	0.74	0.66	0.43	17205

- The total number of low risk (17104) is so high compared to the total number of high risk (101).
- Model accuracy is 0.66 or 66%, since the data in unbalanced we need to look into precision and sensitivity.
- Precision for high risk is extremely low (0.01) which reflect the low number of True Positive high risk (75) corresponding to the total number of positive low and high risk of our model gets (TP+FP = 75+7016 = 7091). However, the sensitivity of detecting true high risks is 0.74 (TP/TP+FP = 75/101=0.74), which means 26 cases of 101 high risk were not detected.
- Sensitivity for low risk is 0.59 which means that 10088 are actually low risk while 17104 cases were detected low risk.

Second Model: SMOTE Oversampling Model
Fig. 2 - SMOTE Oversampling Evaluation



```
In [20]:
# Calculated the balanced accuracy score
y_pred2 = smote_model.predict(X_test)
balanced_accuracy_score(y_test, y_pred2)

Out[20]: 0.6553782219896451

In [21]:
# Display the confusion matrix
confusion_matrix(y_test, y_pred2)

Out[21]: array([[   64,    37],
                [ 5523, 11581]])

In [22]:
# Print the imbalanced classification report
print(classification_report_imbalanced(y_test, y_pred2))
```

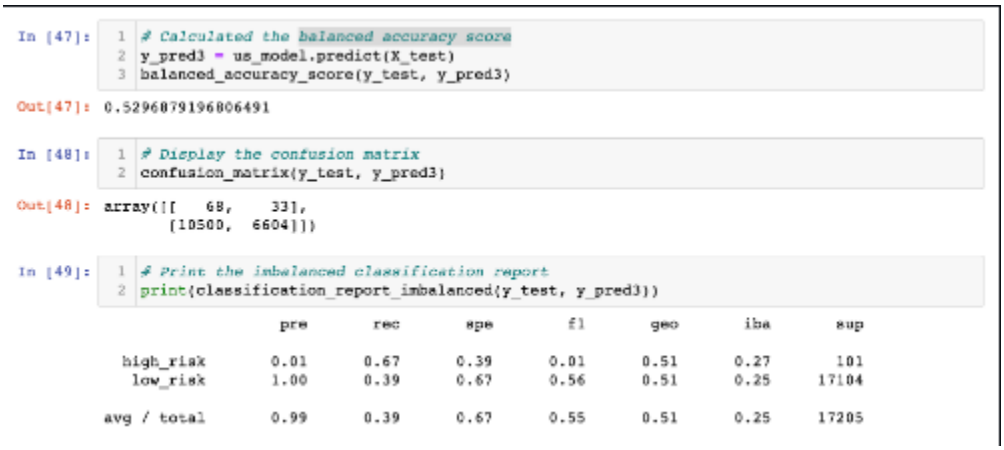
	pre	rec	spe	f1	geo	iba	sup
high_risk	0.01	0.63	0.68	0.02	0.66	0.43	101
low_risk	1.00	0.68	0.63	0.81	0.66	0.43	17104
avg / total	0.99	0.68	0.63	0.80	0.66	0.43	17205

- The total number of negligible risk (17104) is so high compared to the total number of high risk (101).
- Model accuracy is 0.65 or 65%,which is slightly less than that of Naive Random Oversampling Model.
- Since the data in unbalanced we need to look into precision and sensitivity.
- Precision for high risk is still extremely low (0.01) which reflect the low number of True Positive high risk (64) corresponding to the total number of positive low and high risk of our model gets (TP+FP = 64+5523 = 5587). However, the sensitivity of detecting true high risks has decreased to 0.63 (TP/TP+FP = 64/101=0.63), which means 37 cases of 101 high risk were not detected.
- Sensitivity for low risk has increased to 0.68, which means that 5523 cases are actually low risk while they have been detected as high risk.

Naive Random Oversampling Model vs SMOTE:

It would better to adapt the Naive Random Oversampling Model as the sensitivity to detect the high-risk cases is bigger in this model that in SMOTE algorithm. Even though, the sensitivity of detecting low risk is larger in the SMOTE model, however it is better not to miss high-risk cases.

Third Model: Undersampling Model
Fig. 3 - Undersampling Evaluation



```
In [47]:
# Calculated the balanced accuracy score
y_pred3 = us_model.predict(X_test)
balanced_accuracy_score(y_test, y_pred3)

Out[47]: 0.5296879196806491

In [48]:
# Display the confusion matrix
confusion_matrix(y_test, y_pred3)

Out[48]: array([[   68,    33],
                [10500,  6604]])

In [49]:
# Print the imbalanced classification report
print(classification_report_imbalanced(y_test, y_pred3))
```

	pre	rec	spe	f1	geo	iba	sup
high_risk	0.01	0.67	0.39	0.01	0.51	0.27	101
low_risk	1.00	0.39	0.67	0.56	0.51	0.25	17104
avg / total	0.99	0.39	0.67	0.55	0.51	0.25	17205

- Model accuracy in the Undersampling model has decreased to 0.53 or 53%,which is less than both Oversampling Models.
- Since the data in unbalanced we need to look into precision and sensitivity.

- Precision for high risk is still extremely low (0.01) which reflects the low number of True Positive high risk (68) corresponding to the total number of positive low and high risk of our model gets (TP+FP = 68+ 10500 = 5587). However, the sensitivity of detecting true high risks is 0.67 (TP/TP+FP = 68/101=0.67), which means 33 cases of 101 high risk were not detected in this model.
- Sensitivity for low risk has decreased to 0.39, which means that 10500 cases are actually low risk while they have been detected as high risk.
-

Undersampling vs Oversampling:
Even though detecting high risk is more important than detecting low risk. The undersampling algorithm is predicting 10500 from 17104 as being high risk while there are low risk cases. This will put more work on employees to re-study this huge number of False positive high risk loans. Moreover, the balanced accuracy score of this model is the lower than both oversampling algorithms, which is another reason to rule this model out.

Fourth Model: Combination (Over and Under) Sampling
Fig. 4 - Combination Sampling Evaluation

```
In [54]:
# Calculated the balanced accuracy score
y_pred4 = com_model.predict(X_test)
balanced_accuracy_score(y_test, y_pred4)

Out[54]: 0.6499938639794756

In [55]:
# Display the confusion matrix
confusion_matrix(y_test, y_pred4)

Out[55]: array([[   75,   26],
                [7570, 9534]])

In [56]:
# Print the imbalanced classification report
print(classification_report_imbalanced(y_test, y_pred4))
```

	pre	rec	spe	f1	geo	iba	sup
high_risk	0.01	0.74	0.56	0.02	0.64	0.42	101
low_risk	1.00	0.56	0.74	0.72	0.64	0.41	17104
avg / total	0.99	0.56	0.74	0.71	0.64	0.41	17205

- Model accuracy in the Combination Sampling model is to 0.64 or 64%, which is so close to that of both Oversampling Models.
- Since the data in unbalanced we need to look into precision and sensitivity.
- Precision for high risk is still extremely low (0.01) which reflects the low number of True Positive high risk (75) corresponding to the total number of positive low and high risk of our model gets (TP+FP = 75+ 7570 = 7645). However, the sensitivity of detecting true high risks is 0.74 (TP/TP+FP = 75/101=0.74), which means 26 cases of 101 high risk were not detected in this model.
- Sensitivity for low risk is 0.56, which means that 7570 cases are actually low risk while they have been detected as high risk.

Combination (Over and Under) Sampling vs Oversampling Sampling:
Precision for high risk in the combination model (0.74) is higher than that in SMOTE model (0.63) which indicated that this model is detecting more high-risk cases. However, both combination model and random oversampling model have the same precision for high risk (0.74). In this case, one could look in the precision for low risk. In the random oversampling model, it's 0.59 while its 0.56 in the combination model. Therefore, the random oversampling model is detecting less false positive cases and would be the best algorithm among the four.

Fifth Model: Balanced Random Forest Classifier
Fig. 5 - Balanced Random Forest Classifier Evaluation

```
In [26]:
# Calculated the balanced accuracy score
y_pred = brf_model.predict(X_test)
balanced_accuracy_score(y_test, y_pred)

Out[26]: 0.7985836791115968

In [28]:
# Display the confusion matrix
confusion_matrix(y_test, y_pred)

Out[28]: array([[   72,    29],
                [ 1979, 15125]])

In [29]:
# Print the imbalanced classification report
print(classification_report_imbalanced(y_test, y_pred))
```

	pre	rec	spe	f1	geo	iba	sup
high_risk	0.04	0.71	0.88	0.07	0.79	0.62	101
low_risk	1.00	0.88	0.71	0.94	0.79	0.64	17104
avg / total	0.99	0.88	0.71	0.93	0.79	0.64	17205

- Model accuracy in the balanced forest classifier has increased to 0.79 or 79%, which is higher than all previous models.
- Since the data in unbalanced we need to look into precision and sensitivity.
- Precision for high risk is slightly bigger than previous models but it's still low (0.04) which reflects the low number of True Positive high risk (72) corresponding to the total number of positive low and high risk of our model gets (TP+FP = 72+1979 = 2051). Moreover, the sensitivity of detecting true high risks is 0.71 (TP/TP+FP = 72/101=0.71), which means 29 cases of 101 high risk were not detected in this model.
- Sensitivity for low risk has increased to 0.88, which means that 1979 cases are actually low risk while they have been detected as high risk. This number is less than what we got in previous models.

Sixth Model: Easy Ensemble AdaBoost Classifier

Fig. 6- Easy Ensemble AdaBoost Classifier Evaluation

```
In [18]:
# Calculated the balanced accuracy score
y_pred_eec = eec_model.predict(X_test)
balanced_accuracy_score(y_test, y_pred_eec)

Out[18]: 0.9320400994729969

In [19]:
# Display the confusion matrix
confusion_matrix(y_test, y_pred_eec)

Out[19]: array([[   93,     8],
                [  970, 16134]])

In [20]:
# Print the imbalanced classification report
print(classification_report_imbalanced(y_test, y_pred_eec))
```

	pre	rec	spe	f1	geo	iba	sup
high_risk	0.09	0.92	0.94	0.16	0.93	0.87	101
low_risk	1.00	0.94	0.92	0.97	0.93	0.87	17104
avg / total	0.99	0.94	0.92	0.97	0.93	0.87	17205

- Model accuracy in the easy ensemble AdaBoost has increased to 0.93 or 93%, which is the highest value of all models.
- Since the data in unbalanced we need to look into precision and sensitivity.
- Precision for substantial risk has increased to 0.09. However, this low value reflects the low number of True Positive high risk (93) corresponding to the total number of positive low and high risk of our model gets (TP+FP = 93+970=1063). Moreover, the sensitivity of detecting true high risks has increased to 0.92 (TP/TP+FP = 93/101=0.92), which means only 8 cases of 101 high risk were not detected in this model.
- Sensitivity for low risk has also increased to 0.94, which means that only 970 cases are actually low risk while they have been detected as high risk. This number is less than what we got in all previous models.

Conclusion:
Based on all the above, I would recommend using the Easy Ensemble AdaBoost Classifier algorithm because:

- It has the highest accuracy score.
- It has the highest sensitivity of detecting true high risks; Which means only 8 cases with high risks were not detected using this model.
- It has the highest sensitivity for low risk; Which means only 970 cases of 17104 were detected as high risk while in fact they are low risk.