**Data Science Final Project Group 6**

**Presentation Notes: *Machine Learning Model***

Since our dataset contains labeled data and we knew our output, whether the travel insurance was claimed (1) or not (0), we worked with the classification model of supervised machine learning. The classification model's algorithms attempt to learn patterns from the data, and if the model is successful, will gain the ability to make accurate predictions for likelihood of claims.

The dataset was divided into a target and features. The target is the Claim column. The features are all the other columns, because we want to predict whether the person will claim a travel insurance based on different criteria (travel destination, duration, age of insured person, etc.).

We used Scikit-learn library to implement a machine learning model in Python. Logistic Regression was chosen since this is a popular classification model. It predicts binary outcomes, meaning that there are only two possible outcomes. The dataset was split into training and testing sets for the supervised learning model. The dataset was split using train\_test\_split module with 75% of training data and 25% of testing data. The Logistic Regression model learned patterns from the training dataset. It then used the testing dataset to assess its performance.

Logistics Regression itself gave an accuracy of 98.53% and balanced accuracy score of 50%. This huge difference is because the dataset is not balanced. We can see this also from checking the balance of our target values that show us 61,373 lines of data for not claimed insurance and only 917 lines of data for claimed.

Ensemble decision tree models were then used to see how the accuracy scores could be evaluated further to take into consideration the class imbalance, as well as to take into consideration precision and recall. The concept of ensemble learning is the process of combining multiple models, like decision tree algorithms, to help improve the accuracy and robustness, as well as decrease variance of the model, and therefore increase the overall performance of the model.

The following models were used for testing:

* RandomForest
* Adaptive Boosting
* Random Oversampling
* SMOTE
* Random Undersampling
* Cluster Centroid
* SMOTEENN

At first run, as a preprocessing step, we used get\_dummies encoder. The Balanced Random Forest Classifier, Naive Random Oversampling, and Undersampling models were all considered since they provided the highest scores. Undersampling was then ruled out, because although it had a high Balanced Accuracy score, the Recall score was the lowest.

Table

Description automatically generated

The models were then re-run trying OneHotEncoder and LabelEncoder methods, which lowered the scores. The best results were obtained when get\_dummies was applied for all models. Using get\_dummies converts categorical data into dummy or indicator variables. Standard scaler was then applied, and the data was re-run. Variables that are measured at different scales do not contribute equally to the model fitting and model learned function which might end up creating a bias. To deal with this potential problem StandardScaler() module was applied prior to model fitting. Standard scaler improved the scores even more.

Graphical user interface, table, Excel

Description automatically generated

Looking at these results RandomForest shows recall of 0.79, however Accuracy is only 71.92% and the difference between Accuracy and Balanced Accuracy is higher versus other models. The accuracy score is lower than the threshold of 75%. The Balanced Accuracy score for Random Oversampling and recall decreased from 76.7% to 76.3 and 0.75 to 0.73 respectively.

The SMOTE model gave the best results at this point. The original SMOTE Balanced Accuracy score was 72.7% and went up to 76.5% after get\_dummies and standard scaling was applied. Accuracy, Precision, and Recall scores for SMOTE were then 78.2%, 0.05, and 0.75 respectively. The SMOTE model was chosen as our model based on these improved scores.

Table

Description automatically generated

The benefit of using SMOTE is that, unlike oversampling which is random, with SMOTE new instances are interpolated. New values that are created are based on values of its closest neighbors.

A limitation to consider when using SMOTE is that, although it reduces the risk of oversampling, it does not always outperform random oversampling. SMOTE also has a vulnerability to outliers and of course sampling techniques cannot overcome the deficiencies of the original dataset.