CATCHING THE FRAUDSTERS!!!

Project 2: Chelsea Shelton, Manuela Nkwinkwa, Joseph Kuitche

Project Presentation

What are we doing? Catching Fraudulent **vehicle claims** through machine learning predictions

How does it relate to Fintech? delivering better financial services through new technology

Why? Predicting fraud could help save the company money that could be used to provide better services

Objective

Predict Fraudulent claims Through Machine Learning

Selected Models

Supervised: Logistic Regression, KNN, SVM

Dimensionality Reduction: Principal Component Analysis

Deep neural network

[25]:
```
# Seperate the numerical columns and concat with encoded features
numerical_variables = oracle_data[['Deductible', 'AgeOfVehicle', 'DriverRating', 'FraudFound_P']]
encoded_oracle = pd.concat([encoded_variables, numerical_variables], axis=1)
encoded_oracle.dropna(inplace=True)
encoded_oracle.info()
```

[18]:
```
# Create a OneHotEncoder instance
enc = OneHotEncoder(sparse=False)
```

[19]:
```
# Create a list of the columns with categorical variables
categorical_variables = ['AccidentArea', 'Sex', 'MaritalStatus', 'Fault', 'VehicleCategory', 'VehiclePrice',
                         'PastNumberOfClaims', 'AgeOfPolicyHolder', 'PoliceReportFiled', 'WitnessPresent',
                         'AgentType', 'AddressChange_Claim', 'NumberOfCars', 'BasePolicy', 'DayOfWeekClaimed',
                         'MonthClaimed', 'WeekOfMonthClaimed']

# Use the fit_transform method from the OneHotEncoder to encode the categorical variables
encoded_data = enc.fit_transform(oracle_data[categorical_variables])
```

[24]:
```
# Create a DataFrame with the encoded variables
encoded_variables = pd.DataFrame(encoded_data,
    columns = enc.get_feature_names_out(categorical_variables))

encoded_variables.head()
```

Data Preparation & Model Training Process

Data **Sources**: www.kaggle.com

Dimensionality reduction: day of week, Age.

Training process: **over sampling, under sampling**

Performance Evaluation Technique

Demonstration of Machine Learning Model

Unanticipated Insights and adjustment

Our Approach

```
Classifiction Report - Original Data
              precision    recall  f1-score   support

         0.0       0.94      1.00      0.97      3640
         1.0       1.00      0.00      0.01       215

    accuracy                           0.94      3855
   macro avg       0.97      0.50      0.49      3855
weighted avg       0.95      0.94      0.92      3855

---------
Classifiction Report - Undersampled Data
              precision    recall  f1-score   support

         0.0       0.95      0.54      0.69      3640
         1.0       0.07      0.54      0.12       215

    accuracy                           0.54      3855
   macro avg       0.51      0.54      0.40      3855
weighted avg       0.90      0.54      0.66      3855

---------
Classifiction Report - Oversampled Data
              precision    recall  f1-score   support

         0.0       0.95      0.60      0.74      3640
         1.0       0.07      0.49      0.12       215

    accuracy                           0.59      3855
   macro avg       0.51      0.54      0.43      3855
weighted avg       0.90      0.59      0.70      3855
```

Model Results: Logistic Regression

Recall on under sample was the highest to be achieved

Models Evaluation Visuals:KNN & SVM

```
# Create the estimators using dictionary key-value pairs
estimators = {
    'KNeighborsClassifier': knn,
    'SVC': SVC(kernel='rbf', C=10, gamma=0.001)}

# Executing the models
for estimator_name, estimator_object in estimators.items():
    kfold = KFold(n_splits=10, random_state=11, shuffle=True)
    scores = cross_val_score(estimator=estimator_object, X=X_test_scaled, y=y_test, cv=kfold)
    print(f'{estimator_name:>20}: ' +
          f'mean accuracy={scores.mean():.2%}: ' +
          f'standard deviation={scores.std():.2%}')
```

```
KNeighborsClassifier: mean accuracy=94.09%: standard deviation=1.14%
                 SVC: mean accuracy=94.09%: standard deviation=1.14%
```

Model Evaluation: Neural Network

```
#evaluating model

nn.evaluate(X_test_scaled, y_test, verbose=2)
```

```
121/121 - 1s - loss: 0.0564 - accuracy: 0.9351 - 617ms/epoch - 5ms/step
[0.056371293962001 8, 0.9351491332054138]
```

Model Potential:

what we could have done with more time

- Get more data to improve model's performance
- Perform a more complete features engineering
- Evaluate and compare all models

Thank you!

Questions?