Assignment5

February 9, 2024

1 0.) Import the Credit Card Fraud Data From CCLE

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

[2]:
```
df = pd.read_csv("fraudTest.csv")
```

[3]:
```
df.head()
```

[3]:
```
   Unnamed: 0 trans_date_trans_time            cc_num  \
0           0   2020-06-21 12:14:25  2291163933867244
1           1   2020-06-21 12:14:33  3573030041201292
2           2   2020-06-21 12:14:53  3598215285024754
3           3   2020-06-21 12:15:15  3591919803438423
4           4   2020-06-21 12:15:17  3526826139003047

                              merchant        category    amt   first  \
0                fraud_Kirlin and Sons   personal_care   2.86    Jeff
1                 fraud_Sporer-Keebler   personal_care  29.84  Joanne
2  fraud_Swaniawski, Nitzsche and Welch  health_fitness  41.28  Ashley
3                     fraud_Haley Group        misc_pos  60.05   Brian
4                 fraud_Johnston-Casper          travel   3.19  Nathan

       last gender                       street  …      lat      long  \
0   Elliott      M            351 Darlene Green  …  33.9659  -80.9355
1  Williams      F             3638 Marsh Union  …  40.3207 -110.4360
2     Lopez      F         9333 Valentine Point  …  40.6729  -73.5365
3  Williams      M  32941 Krystal Mill Apt. 552  …  28.5697  -80.8191
4    Massey      M     5783 Evan Roads Apt. 465  …  44.2529  -85.0170

   city_pop                        job         dob  \
0    333497        Mechanical engineer  1968-03-19
1       302  Sales professional, IT  1990-01-17
2     34496          Librarian, public  1970-10-21
3     54767              Set designer  1987-07-25
4      1126        Furniture designer  1955-07-06
```

```
                          trans_num   unix_time  merch_lat  merch_long  \
0  2da90c7d74bd46a0caf3777415b3ebd3  1371816865  33.986391  -81.200714
1  324cc204407e99f51b0d6ca0055005e7  1371816873  39.450498 -109.960431
2  c81755dbbbea9d5c77f094348a7579be  1371816893  40.495810  -74.196111
3  2159175b9efe66dc301f149d3d5abf8c  1371816915  28.812398  -80.883061
4  57ff021bd3f328f8738bb535c302a31b  1371816917  44.959148  -85.884734

   is_fraud
0         0
1         0
2         0
3         0
4         0

[5 rows x 23 columns]
```

[6]:
```
df_select = df[["trans_date_trans_time", "category", "amt", "city_pop",␣
 ↪"is_fraud"]].copy()

df_select["trans_date_trans_time"] = pd.
 ↪to_datetime(df_select["trans_date_trans_time"])


df_select["time_var"] = df_select["trans_date_trans_time"].dt.second


X = pd.get_dummies(df_select, columns=["category"]).
 ↪drop(["trans_date_trans_time", "is_fraud"], axis=1)
y = df_select["is_fraud"]
```

2 1.) Use scikit learn preprocessing to split the data into 70/30 in out of sample

[7]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

[8]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .3)
```

[9]:
```
X_test, X_holdout, y_test, y_holdout = train_test_split(X_test, y_test,␣
 ↪test_size = .5)
```

[10]:
```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
```

[11]:
```
X_test = scaler.transform(X_test)
X_holdout = scaler.transform(X_holdout)
```

3 2.) Make three sets of training data (Oversample, Undersample and SMOTE)¶

[12]:
```
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import SMOTE
```

[13]:
```
ros = RandomOverSampler()
over_X, over_y = ros.fit_resample(X_train, y_train)

rus = RandomUnderSampler()
under_X, under_y = rus.fit_resample(X_train, y_train)

smote = SMOTE()
smote_X, smote_y = smote.fit_resample(X_train, y_train)
```

4 3.) Train three logistic regression models

[14]:
```
from sklearn.linear_model import LogisticRegression
```

[15]:
```
over_log = LogisticRegression().fit(over_X, over_y)

under_log = LogisticRegression().fit(under_X, under_y)

smote_log = LogisticRegression().fit(smote_X, smote_y)
```

5 4.) Test the three models

[16]:
```
over_log.score(X_test, y_test)
```

[16]: 0.9186160896374673

[17]:
```
under_log.score(X_test, y_test)
```

[17]: 0.9240744739557091

[18]:
```
smote_log.score(X_test, y_test)
```

[18]: 0.9154850164351352

We see SMOTE performing with higher accuracy but is ACCURACY really the best measure?

6 5.) Which performed best in Out of Sample metrics?

[19]:
```
from sklearn.metrics import confusion_matrix
```

[20]:
```
y_true = y_test
```

[21]:
```
y_pred = over_log.predict(X_test)
cm = confusion_matrix(y_true, y_pred)
cm
```

[21]:
```
array([[76363,  6696],
       [   88,   211]])
```

[22]:
```
print("Over Sample Sensitivity : ", cm[1,1] /( cm[1,0] + cm[1,1]))
```

```
Over Sample Sensitivity :  0.705685618729097
```

[23]:
```
y_pred = under_log.predict(X_test)
cm = confusion_matrix(y_true, y_pred)
cm
```

[23]:
```
array([[76820,  6239],
       [   90,   209]])
```

[24]:
```
print("Under Sample Sensitivity : ", cm[1,1] /( cm[1,0] + cm[1,1]))
```

```
Under Sample Sensitivity :  0.6989966555183946
```

[25]:
```
y_pred = smote_log.predict(X_test)
cm = confusion_matrix(y_true, y_pred)
cm
```

[25]:
```
array([[76102,  6957],
       [   88,   211]])
```

[26]:
```
print("SMOTE Sample Sensitivity : ", cm[1,1] /( cm[1,0] + cm[1,1]))
```

```
SMOTE Sample Sensitivity :  0.705685618729097
```

7 7.) We want to compare oversampling, Undersampling and SMOTE across our 3 models (Logistic Regression, Logistic Regression Lasso and Decision Trees).

8 Make a dataframe that has a dual index and 9 Rows.

9 Calculate: Sensitivity, Specificity, Precision, Recall and F1 score. for out of sample data.

10 Notice any patterns across perfomance for this model. Does one totally out perform the others IE. over/under/smote or does a model perform better DT, Lasso, LR?

11 Choose what you think is the best model and why. test on Holdout

[29]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, precision_score, recall_score,␣
 ↪f1_score
import pandas as pd
from imblearn.over_sampling import RandomOverSampler
```

[30]:
```
resampling_methods = {
    "over": RandomOverSampler(),
    "under": RandomUnderSampler(),
    "smote": SMOTE()
}

model_configs = {
    "LOG":LogisticRegression(),
    "LASSO": LogisticRegression(penalty = "l1", C = 2., solver = "liblinear"),
    "DTREE": DecisionTreeClassifier()
}
```

[47]:
```
def calc_perf_metric(y_true, y_pred):
    tn,fp,fn,tp = confusion_matrix(y_true, y_pred).ravel()

    sensitivity = tp / (tp + fn)
    specificity = tn / (tn + fp)
    precision = precision_score(y_true, y_pred),
    recall = recall_score(y_true, y_pred),
    f1 = f1_score(y_true, y_pred)


    return(sensitivity, specificity, precision, recall, f1)
```

[32]:
```python
trained_models = {}
results = []
```

[48]:
```python
for resample_key, resampler in resampling_methods.items():
    resample_X, resample_y = resampler.fit_resample(X_train, y_train)

    for model_key, model in model_configs.items():
        combined_key = f"{resample_key}_{model_key}"

        m = model.fit(resample_X, resample_y)

        trained_models[combined_key] = m

        y_pred = m.predict(X_test)

        sensitivity, specificity, precision, recall, f1 =␣
 ↪calc_perf_metric(y_test,y_pred)

        results.append({"Model": combined_key,
                        "Sensitivity" : sensitivity,
                        "Specificity": specificity,
                        "Precision": precision,
                        "Recall": recall,
                        "F1" : f1})
```

[49]:
```python
result_df = pd.DataFrame(results)
```

[50]:
```python
result_df
```

[50]:
```
          Model  Sensitivity  Specificity                Precision  \
0      over_LOG     0.705686     0.915205   (0.02908740005514199,)
1    over_LASSO     0.705686     0.915277   (0.02911147902869757,)
2    over_DTREE     0.515050     0.998507    (0.5539568345323741,)
3     under_LOG     0.705686     0.926751    (0.0335186656076251,)
4   under_LASSO     0.705686     0.926257   (0.03330176767676768,)
5   under_DTREE     0.946488     0.949265   (0.06293084278407828,)
6     smote_LOG     0.705686     0.914579  (0.028880372296742403,)
7   smote_LASSO     0.705686     0.914663  (0.028908069598575146,)
8   smote_DTREE     0.692308     0.993198   (0.26813471502590674,)

                  Recall        F1
0   (0.705685618729097,)  0.055872
1   (0.705685618729097,)  0.055916
2  (0.5150501672240803,)  0.533795
3   (0.705685618729097,)  0.063998
4   (0.705685618729097,)  0.063602
5  (0.9464882943143813,)  0.118015
```

```
6   (0.705685618729097,)  0.055490
7   (0.705685618729097,)  0.055541
8  (0.6923076923076923,)  0.386555
```