← Go Back to Feature Selection, Model Selection and Tuning

Course Content

Hands-on Quiz

Type	:	Graded Quiz
Attempts	:	1/1
Questions	:	10
Time	:	2h
Due Date	:	Dec 06, 2021, 11:30 AM
Your Score	:	20/20

Instructions

Attempt History

Attempt #1

Dec 06, 2021, 11:15 AM

Marks: 20

Q No: 1 Correct Answer

Marks: 2/2

Load the cardiac dataset. Identify the correct percentage of the positive (Yes) and negative (No) class distribution.

- Yes: 35% and No: 65%
- Yes: 65% and No: 35%
- Yes: 78.6% and No: 21.4%
- (selected) Yes: 21.4 % and No: 78.6% — You Selected

```
import pandas as pd
df = pd.read_csv('cardiac.csv')
df['UnderRisk'].value_counts(normalize = True)
```

Q No: 2 — Correct Answer

Marks: 2/2

Prepare the data according to the following instructions in a sequential manner.

- Encode target variable (Replace yes with 1 and no with 0)
- Split the data into temp and test in the 80:20 ratio. Use the parameter 'stratify' while splitting the data
- Split the temp set into train and validation in the 75:25 ratio. Use the parameter 'stratify' while splitting the data
- Create dummies for X_train, X_val, and X_test. Use drop_first = True while creating dummies.

Note - Do not do any other data processing. It might lead to a mismatch of the outcomes.

How many rows and columns are there in the train set?

533 rows and 13 columns You Selected

180 rows and 13 columns

178 rows and 13 columns

889 rows and 13 columns

```
## Encoding Existing and Attrited customers to 0 and 1 respectively, for analysis.

df["UnderRisk"].replace("yes", 1, inplace=True)

df["UnderRisk"].replace("no", 0, inplace=True)


X = df.drop(["UnderRisk"], axis=1)

y = df["UnderRisk"]


from sklearn.model_selection import train_test_split

# Splitting data into training, validation and test set:

# first we split data into 2 parts, say temporary and test


X_temp, X_test, y_temp, y_test = train_test_split(

    X, y, test_size=0.2, random_state=1, stratify=y

)

# then we split the temporary set into train and validation

X_train, X_val, y_train, y_val = train_test_split(

    X_temp, y_temp, test_size=0.25, random_state=1, stratify=y_temp
```

```
)

print(X_train.shape, X_val.shape, X_test.shape)



X_train = pd.get_dummies(X_train, drop_first=True)

X_val = pd.get_dummies(X_val, drop_first=True)

X_test = pd.get_dummies(X_test, drop_first=True)

print(X_train.shape, X_val.shape, X_test.shape)
```

Q No: 3 Correct Answer

Marks: 2/2

Here, we are attempting to predict whether a person will have cardiac arrest or not based on his medical background. According to you, which of the following would be the most appropriate metric of evaluation for prediction.

- ○ Accuracy
- ○ Precision
- ◉ Recall — You Selected
- ○ F1 score

In medical cases, predicting the presence of disease in the patients who actually have a disease is more important than predicting the absence of the disease in the patients who do not have the disease. In such cases, reducing the number of FP is more important than FN. Therefore, the right metric would be recall.

Q No: 4 Correct Answer

Marks: 2/2

Now that we have decided on our evaluation metric, we can go ahead to build models. Since cardiac arrest prediction is a classification problem, we can start with logistic regression.

Train the models as per the following instructions.

- Build a logistic regression on the train set using the sklearn implementation with default parameters and random_state=1 and check the performance of the model on the train set.
- Oversample the train set using SMOTE with parameters listed below, build a logistic regression on the oversampled data, and check the performance of the model on the oversampled train set.

SMOTE parameters: sampling_strategy=1, k_neighbors=5, random_state=1

Which of the following statements is true on comparing the performance of both the models on train data and oversampled train data.

- (●) Accuracy of the model on an oversampled set has decreased whereas recall and precision of the oversampled data have increased. — You Selected
- ( ) Accuracy of the model on an oversampled set has increased whereas recall and precision of the oversampled data have decreased.
- ( ) Accuracy, precision, and recall of the oversampled data have decreased.
- ( ) Accuracy and recall of the model on an oversampled set have decreased whereas the precision of the oversampled data has increased.

```
from sklearn.linear_model import LogisticRegression

lr1 = LogisticRegression(random_state=1)

lr1.fit(X_train, y_train)

model_performance_classification_sklearn(lr1, X_train, y_train)
```

```
sm = SMOTE(
    sampling_strategy=1, k_neighbors=5, random_state=1
)  # Synthetic Minority Over Sampling Technique
X_train_over, y_train_over = sm.fit_resample(X_train, y_train)
lr2 = LogisticRegression(random_state=1)
lr2.fit(X_train_over, y_train_over)
model_performance_classification_sklearn(lr2, X_train_over, y_train_over)
```

Q No: 5 Correct Answer

Marks: 2/2

Let's try some other algorithms before settling on a final model. Train a bagging classifier on the oversampled data using the sklearn implementation with default parameters and random_state=1 and check the model performance on the validation set.

Which of the following options gives the correct range of the evaluation metrics?

Recall: In a range of 0.55 to 0.65

Accuracy: In a range of 0.65 to 0.75

Recall: In a range of 0.45 to 0.55

Precision: In a range of 0.35 to 0.45

You Selected

Recall: In a range of 0.55 to 0.65

Precision: In a range of 0.20 to 0.30

Recall: In a range of 0.45 to 0.55

F1 score: In a range of 0.30 to 0.40

```
from sklearn.ensemble import BaggingClassifier

bag = BaggingClassifier(random_state=1)

bag.fit(X_train_over, y_train_over)

model_performance_classification_sklearn(bag, X_val, y_val)
```

Q No: 6 Correct Answer

Marks: 2/2

Let's try one more model and see how our model performs. Along with the evaluation metrics, we should check how many observations are correctly predicted by our model.

Train a random forest classifier with the original training set using the sklearn implementation with default parameters and random_state=1 and assess the model performance on the training

set.

Identify the correct range of number of cases that are correctly predicted as 'yes' by the random forest classifier:

- 30 to 40
- 15 to 25 — You Selected
- 0 to 5
- 8 to 13

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(random_state=1)

rf.fit(X_train, y_train)

confusion_matrix(y_train, rf.predict(X_train))
```

Q No: 7 Correct Answer

Marks: 2/2

One model might not give the desired outcome, we can try different models and compare their performances. Let's try different models.

Train the models as per the following instructions.

- Train Bagging classifier using BaggingClassifier(random_state=1)
- Train Random forest classifier using RandomForestClassifier(random_state=1)
- Train Logistic regression using LogisticRegression(random_state=1)
- Train Decision trees using DecisionTreeClassifier(random_state=1)

- Loop through all the above models to get the mean cross-validated scores. Use the following code for the CV results on over sampled data -

```
scoring = "recall"
  kfold = StratifiedKFold(
    n_splits=5, shuffle=True, random_state=1
  ) # Setting number of splits equal to 5
  cv_result = cross_val_score(
    estimator=model, X=X_train_over, y=y_train_over, scoring=scoring, cv=kfold
  )
```

Which of the following statements are true about the cross-validated recall scores on the oversampled data?

A. The average cross-validated recall score for the bagging classifier and the random forest is approximately the same.

B. The difference between the CV recall scores for logistic regression and decision tree is in the range of 1-5.

C. The CV recall score for logistic regression lies in the range of 0.75 to 0.90

D. The CV recall score for decision trees lies in the range of 0.75 to 0.90

- ○ A, B and C
- ◉ A, C and D — You Selected
- ○ A and C
- ○ B, C and D

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import StratifiedKFold, cross_val_score
```

```
models = []  # Empty list to store all the models

# Appending models into the list

models.append(("Bagging", BaggingClassifier(random_state=1)))

models.append(("Random forest", RandomForestClassifier(random_state=1)))

models.append(("LR", LogisticRegression(random_state=1)))

models.append(("dtree", DecisionTreeClassifier(random_state=1)))

results = []  # Empty list to store all model's CV scores

names = []  # Empty list to store name of the models

# loop through all models to get the mean cross validated score

print("\n" "Cross-Validation Performance:" "\n")

for name, model in models:

    scoring = "recall"

    kfold = StratifiedKFold(

        n_splits=5, shuffle=True, random_state=1

    )  # Setting number of splits equal to 5

    cv_result = cross_val_score(

        estimator=model, X=X_train_over, y=y_train_over, scoring=scoring, cv=kfold

    )

    results.append(cv_result)

    names.append(name)

    print("{}: {}".format(name, cv_result.mean() * 100))
```

Q No: 8 Correct Answer

Marks: 2/2

Building the model with default parameters might not give a satisfactory outcome. Let's try to identify the best combination of the hyperparameters.

Train an AdaBoost classifier using the oversampled data and tune the model using random search.

Use the following code to define the parameters -

```
param_grid = {
    "n_estimators": np.arange(10, 110, 10),
    "learning_rate": [0.1, 0.01, 0.2, 0.05, 1],
    "base_estimator": [
        DecisionTreeClassifier(max_depth=1, random_state=1),
        DecisionTreeClassifier(max_depth=2, random_state=1),
        DecisionTreeClassifier(max_depth=3, random_state=1),
    ],
}
# Type of scoring used to compare parameter combinations
scorer = metrics.make_scorer(metrics.recall_score)
# Calling RandomizedSearchCV
randomized_cv = RandomizedSearchCV(
    estimator=model,
    param_distributions=param_grid,
    n_jobs=-1,
    n_iter=50,
    scoring=scorer,
    cv=5,
    random_state=1,
)
```

Which of the following is the best combination of the hyperparameters obtained on tuning the Adaboost classifier with oversampled data?

- Best combination of the parameters are {'n_estimators': 50, 'learning_rate': 0.1, 'base_estimator': DecisionTreeClassifier(max_depth=1, random_state=1)}
- Best combination of the parameters are {'n_estimators': 50, 'learning_rate': 0.01, 'base_estimator': DecisionTreeClassifier(max_depth=2, random_state=2)}
- Best combination of the parameters are {'n_estimators': 70, 'learning_rate': 0.01, 'base_estimator': DecisionTreeClassifier(max_depth=1, random_state=1)} — You Selected
- Best combination of the parameters are {'n_estimators': 40, 'learning_rate': 0.001, 'base_estimator': DecisionTreeClassifier(max_depth=1, random_state=1)}

```
from sklearn.ensemble import AdaBoostClassifier

from sklearn import metrics

from sklearn.model_selection import RandomizedSearchCV


# defining model

model = AdaBoostClassifier(random_state=1)

# Parameter grid to pass in GridSearchCV

param_grid = {

    "n_estimators": np.arange(10, 110, 10),

    "learning_rate": [0.1, 0.01, 0.2, 0.05, 1],

    "base_estimator": [

        DecisionTreeClassifier(max_depth=1, random_state=1),

        DecisionTreeClassifier(max_depth=2, random_state=1),
```

```
        DecisionTreeClassifier(max_depth=3, random_state=1),
    ],
}
# Type of scoring used to compare parameter combinations
scorer = metrics.make_scorer(metrics.recall_score)
# Calling RandomizedSearchCV
randomized_cv = RandomizedSearchCV(
    estimator=model,
    param_distributions=param_grid,
    n_jobs=-1,
    n_iter=50,
    scoring=scorer,
    cv=5,
    random_state=1,
)
# Fitting parameters in RandomizedSearchCV
randomized_cv.fit(X_train_over, y_train_over)
print(
    "Best parameters are {} with CV score={}:".format(
        randomized_cv.best_params_, randomized_cv.best_score_
    )
)
```

Q No: 9 Correct Answer

Marks: 2/2

We can further check how our model performs on oversampled and undersampled data.

Train the Adaboost classifier with undersampled and oversampled data. Assess the model performance for Adaboost with oversampled data on the oversampled train data and for Adaboost with undersampled data on the undersampled train data.

Which of the following statements is true about the performance of the model?

- ○ The performance of the model trained with undersampled data is similar to the training performance of the model trained with oversampled data.
- ◉ The performance of the model trained with undersampled data is better than the performance of the model trained with oversampled data. — You Selected
- ○ The performance of the model trained with oversampled data is better than the performance of the model trained with undersampled data.

```
from imblearn.under_sampling import RandomUnderSampler

rus = RandomUnderSampler(random_state=1)

X_train_un, y_train_un = rus.fit_resample(X_train, y_train)

model1 = AdaBoostClassifier(random_state=1)

model1.fit(X_train_un, y_train_un)

model_performance_classification_sklearn(model1, X_train_un, y_train_un)

model2 = AdaBoostClassifier(random_state=1)

model2.fit(X_train_over, y_train_over)

model_performance_classification_sklearn(model2, X_train_over, y_train_over)
```

Q No: 10 Correct Answer

Marks: 2/2

It is important to understand the features that are critical in making the right predictions. Let's try out what are the important features for our model.

Plot the feature importance of the variables for the Adaboost classifier trained with undersampled data.

Which of the following are the most important features?

- ○ Use_of_stimulant_drugs and Obese
- ◉ Gender and HighBP — You Selected
- ○ Gender and Obese
- ○ Use_of_stimulant_drugs and HighBP

```
import matplotlib.pyplot as plt

model1 = AdaBoostClassifier(random_state=1)

model1.fit(X_train_un, y_train_un)

feature_names = X_train_un.columns

importances = model1.feature_importances_

indices = np.argsort(importances)

plt.figure(figsize=(12, 12))

plt.title("Feature Importances")

plt.barh(range(len(indices)), importances[indices], color="violet", align="center")

plt.yticks(range(len(indices)), [feature_names[i] for i in indices])

plt.xlabel("Relative Importance")

plt.show()
```

Comments:

+ Add comments

Previous

Next