08

A importância dos países

Observamos no projeto que o país de onde o cliente tentou efetuar a compra não parece ser um fator importante. Entretanto, seu chefe gostaria que você testasse um modelo preditivo com os países para ter certeza de que eles não ajudariam no resultado. Sendo assim, importe um banco de dados contendo os países como variável dummy e também as variáveis media_id e tempo. Então, vamos adotar o mesmo protocolo que seguimos para os modelos sem a variável país. Divida os dados da mesma forma que fizemos durante o curso:

```
X_dev, X_eval, Y_dev, Y_eval = train_test_split(X, Y, train_size=.75, random_state=23)
```

Adicione o vetor de destino de volta ao conjunto de recursos:

```
["fraude"] = Y
```

Separe as anomalias dos itens normais:

```
X_anomalies = X[ X["fraude"] == 1 ]
X_normals = X[ X["fraude"] == 0 ]
```

E separe uma amostra aleatória de 70% dos índices do conjunto de dados normal para treinamento, guardando o resto para o conjunto de testes:

```
train_idxs = sample(list(X_normals.index), int(0.7 * X_normals.shape[0]))
X_train = X_normals.loc[train_idxs]
X_testing = X_normals.drop(train_idxs)
```

O conjunto de testes deve agora ser a combinação das instâncias normais restantes, que não estão no conjunto de treinamento, e as instâncias de anomalia:

```
X_testing = pd.concat([X_testing, X_anomalies], axis=0)
```

Podemos embaralhar as linhas para ambos os conjuntos de dados:

```
X_train = X_train.sample(frac=1).reset_index(drop=True)
X_testing = X_testing.sample(frac=1).reset_index(drop=True)
```

Divida o dataframe de teste em conjuntos de dados de validação cruzada e de avaliação:

```
Y_testing = X_testing["fraude"]
X_testing = X_testing[ [col for col in X_testing.columns if col != "fraude"] ]
X_cv, X_eval, Y_cv, Y_eval = train_test_split(X_testing, Y_testing, train_size=.7, random_state
```

Mapeie os valores `Y_cv` e `Y_val` para a convenção 1 (normal), -1 (anomalias):

```
Y_cv = Y_cv.apply(lambda x: 1 if x == 0 else -1)
Y_eval = Y_eval.apply(lambda x: 1 if x == 0 else -1)
```

Exclua a coluna de classe do conjunto de treinamento:

```
X_train = X_train[ [col for col in X_testing.columns if col != "fraude"] ]
```

Faça um reset dos índices e exclua a flag de fraude:

```
X_train = X_train.sample(frac=1).reset_index(drop=True)
X_testing = X_train.sample(frac=1).reset_index(drop=True)
X_dev.drop(["fraude"], axis=1, inplace = True)
```

Finalmente, separe os dados de teste e de treinamento:

```
Y_testing = X_testing['fraude']
X_testing = X_testing[[col for col in X_testing.columns if col!='fraude']]
X_cv, X_eval, Y_cv, Y_eval = train_test_split(X_testing, Y_testing, train_size=0.7, random_state
```

```
Y_cv = Y_cv.apply(lambda x: 1 if x == 0 else -1)
Y_eval = Y_cv.apply(lambda x: 1 if x == 0 else -1)

X_train = X_train[[col for col in X_testing.columns if col!= 'fraude']]
```

Agora que seus dados estão prontos, crie um algoritmo de árvores de decisão que calcule as probabilidades de um cliente ser ou não um fraudador para apresentar ao seu chefe.