Seleção de atributos

Neste código utilizamos o método de seleção de atributos, que é crucial na descoberta de conhecimento e na mineração de dados. Nem todos os atributos em um conjunto de dados são igualmente importantes para a sua análise. A seleção de atributos ajuda a: Reduzir a dimensionalidade dos dados: Isso pode melhorar a eficiência e a precisão dos algoritmos de mineração de dados.

1. Treinamento do modelo com todos os atributos:

```
# treinando e avaliando o modelo com os atributos selecionados
clf = DecisionTreeClassifier(random_state=1)
clf.fit(X_train_top, y_train)
y_pred_selected = clf.predict(X_test_top)
```

Nessa parte, o modelo considera todos os atributos, o que pode incluir dados irrelevantes que podem prejudicar o desempenho, especialmente em dados desbalanceados.

2. Seleção de atributos (SelectKBest):

```
# selecionando os 10 melhores atributos com base no teste F
selector = SelectKBest(score_func=f_classif, k=10)
X_train_selected = selector.fit_transform(X_train, y_train)
X_test_selected = selector.transform(X_test)

# exibindo as pontuações de cada atributo
scores = selector.scores_
feature_scores = list(zip(X.columns, scores))
feature_scores.sort(key=lambda x: x[1], reverse=True)  # ordenando atributos por relevância

print("Pontuação dos atributos:")
for feature, score in feature_scores:
    print(f"{feature}: {score}")
```

Utilizamos o SelectKBest para selecionar os 10 melhores atributos com base na sua correlação com a variável alvo, usando o teste F (ANOVA F-test). As pontuações de cada atributo são exibidas, com isso a seleção de atributos elimina variáveis menos relevantes, ajudando a melhorar a eficiênca do modelo e sua capacidade de generalização, especialmente em dados ruidosos.

3. Treinamento com atributos selecionados:

```
    # treinando e avaliando o modelo com os atributos selecionados
    clf = DecisionTreeClassifier(random_state=1)
    clf.fit(X_train_top, y_train)
    y_pred_selected = clf.predict(X_test_top)

    print("\nDesempenho com atributos selecionados:")
    print("Acurácia:", accuracy_score(y_test, y_pred_selected))
    print("Matriz de Confusão:")
    print(confusion_matrix(y_test, y_pred_selected))
    print("F1 Score:", f1_score(y_test, y_pred_selected))
```

Depois de selecionar os atributos mais relevantes, o modelo é treinado novamente apenas com esses atributos selecionados. O desempenho do modelo é melhorado, focando nas variáveis que mais influenciam a predição de fraudes e reduzindo a complexidade.

https://colab.research.google.com/drive/1TVQeBJiULr0dLsrCJno9UZtixPLaQFaN#scrollTo=XOi-Itn81Eim