Machine_Learning_Validação.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda Salvo pela última vez às 08:07

Comentário Compartilhar

+ Código + Texto Conectar

```
import pandas as pd

uri = "https://gist.githubusercontent.com/guilhermesilveira/e99a526b2e7ccc6c3b70f53db43a87d2/raw/1605fc74aa778066bf2e6695e24d53cf65f2f447/machine-learning-carro
dados = pd.read_csv(uri).drop(columns=["Unnamed: 0"], axis=1)
dados.head()
```

	preco	vendido	idade_do_modelo	km_por_ano
0	30941.02	1	18	35085.22134
1	40557.96	1	20	12622.05362
2	89627.50	0	12	11440.79806
3	95276.14	0	3	43167.32682
4	117384.68	1	4	12770.11290

```
[ ] import numpy as np
    from sklearn.model_selection import train_test_split
    from sklearn.svm import LinearSVC
    from sklearn.metrics import accuracy_score

    x = dados[["preco", "idade_do_modelo","km_por_ano"]]
    y = dados["vendido"]

    SEED = 158020
    np.random.seed(SEED)
    treino_x, teste_x, treino_y, teste_y = train_test_split(x, y, test_size = 0.25,
                                                            stratify = y)
    print("Treinaremos com %d elementos e testaremos com %d elementos" % (len(treino_x), len(teste_x)))
```

```
Treinaremos com 7500 elementos e testaremos com 2500 elementos
```

```
[ ] from sklearn.dummy import DummyClassifier

    dummy_stratified = DummyClassifier()
    dummy_stratified.fit(treino_x, treino_y)
    acuracia = dummy_stratified.score(teste_x, teste_y) * 100

    print("A acurácia do dummy stratified foi %.2f%%" % acuracia)
```

```
A acurácia do dummy stratified foi 50.96%
```

```
[ ] from sklearn.tree import DecisionTreeClassifier

    SEED = 158020
    np.random.seed(SEED)
    modelo = DecisionTreeClassifier(max_depth=2)
    modelo.fit(treino_x, treino_y)
    previsoes = modelo.predict(teste_x)

    acuracia = accuracy_score(teste_y, previsoes) * 100
    print("A acurácia foi %.2f%%" % acuracia)
```

```
A acurácia foi 71.92%
```

```
[ ] x = dados[["preco", "idade_do_modelo","km_por_ano"]]
    y = dados["vendido"]

    SEED = 158020
    np.random.seed(SEED)
    treino_x, teste_x, treino_y, teste_y = train_test_split(x, y, test_size = 0.25,
                                                            stratify = y)
    print("Treinaremos com %d elementos e testaremos com %d elementos" % (len(treino_x), len(teste_x)))

    modelo = DecisionTreeClassifier(max_depth=2)
    modelo.fit(treino_x, treino_y)
    previsoes = modelo.predict(teste_x)

    acuracia = accuracy_score(teste_y, previsoes) * 100
    print("A acurácia foi %.2f%%" % acuracia)
```

```
Treinaremos com 7500 elementos e testaremos com 2500 elementos
A acurácia foi 71.92%
```

```
[ ] from sklearn.model_selection import cross_validate

    SEED = 301
    np.random.seed(SEED)

    modelo = DecisionTreeClassifier(max_depth=2)
    results = cross_validate(modelo, x, y, cv = 3, return_train_score=False)
```

```
media = results['test_score'].mean()
desvio_padrao = results['test_score'].std()
print("Accuracy com cross validation, 3 = [%.2f, %.2f]" % ((media - 2 * desvio_padrao)*100, (media + 2 * desvio_padrao) * 100))
```

```
Accuracy com cross validation, 3 = [74.99, 76.57]
```

```
[ ] SEED = 301
    np.random.seed(SEED)

    modelo = DecisionTreeClassifier(max_depth=2)
    results = cross_validate(modelo, x, y, cv = 10, return_train_score=False)
    media = results['test_score'].mean()
    desvio_padrao = results['test_score'].std()
    print("Accuracy com cross validation, 10 = [%.2f, %.2f]" % ((media - 2 * desvio_padrao)*100, (media + 2 * desvio_padrao) * 100))
```

```
Accuracy com cross validation, 10 = [74.24, 77.32]
```

```
[ ] SEED = 301
    np.random.seed(SEED)

    modelo = DecisionTreeClassifier(max_depth=2)
    results = cross_validate(modelo, x, y, cv = 5, return_train_score=False)
    media = results['test_score'].mean()
    desvio_padrao = results['test_score'].std()
    print("Accuracy com cross validation, 5 = [%.2f, %.2f]" % ((media - 2 * desvio_padrao)*100, (media + 2 * desvio_padrao) * 100))
```

```
Accuracy com cross validation, 5 = [75.21, 76.35]
```

Aleatoriedade no cross validate

```
[ ] def imprime_resultados(results):
      media = results['test_score'].mean()
      desvio_padrao = results['test_score'].std()
      print("Accuracy médio: %.2f" % (media * 100))
      print("Accuracy intervalo: [%.2f, %.2f]" % ((media - 2 * desvio_padrao)*100, (media + 2 * desvio_padrao) * 100))
```

```
[ ] from sklearn.model_selection import KFold

    SEED = 301
    np.random.seed(SEED)

    cv = KFold(n_splits = 10)
    modelo = DecisionTreeClassifier(max_depth=2)
    results = cross_validate(modelo, x, y, cv = cv, return_train_score=False)
    imprime_resultados(results)
```

```
Accuracy médio: 75.78
Accuracy intervalo: [74.37, 77.19]
```

```
[ ] SEED = 301
    np.random.seed(SEED)

    cv = KFold(n_splits = 10, shuffle = True)
    modelo = DecisionTreeClassifier(max_depth=2)
    results = cross_validate(modelo, x, y, cv = cv, return_train_score=False)
    imprime_resultados(results)
```

```
Accuracy médio: 75.76
Accuracy intervalo: [73.26, 78.26]
```

Simular situação horrível de azar

Pode ser "azar" como pode ser uma proporção de exemplos desbalanceado entre as classes.

```
[ ] dados_azar = dados.sort_values("vendido", ascending=True)
    x_azar = dados_azar[["preco", "idade_do_modelo","km_por_ano"]]
    y_azar = dados_azar["vendido"]
    dados_azar.head()
```

	preco	vendido	idade_do_modelo	km_por_ano
4999	74023.29	0	12	24812.80412
5322	84843.49	0	13	23095.63834
5319	83100.27	0	19	36240.72746
5316	87932.13	0	16	32249.56426
5315	77937.01	0	15	28414.50704

```
[ ] from sklearn.model_selection import KFold

    SEED = 301
    np.random.seed(SEED)
```

```
cv = KFold(n_splits = 10)
modelo = DecisionTreeClassifier(max_depth=2)
results = cross_validate(modelo, x_azar, y_azar, cv = cv, return_train_score=False)
imprime_resultados(results)
```

```
Accuracy médio: 57.84
Accuracy intervalo: [34.29, 81.39]
```

```
[ ] from sklearn.model_selection import KFold

    SEED = 301
    np.random.seed(SEED)

    cv = KFold(n_splits = 10, shuffle=True)
    modelo = DecisionTreeClassifier(max_depth=2)
    results = cross_validate(modelo, x_azar, y_azar, cv = cv, return_train_score=False)
    imprime_resultados(results)
```

```
Accuracy médio: 75.78
Accuracy intervalo: [72.30, 79.26]
```

```
[ ] from sklearn.model_selection import StratifiedKFold

    SEED = 301
    np.random.seed(SEED)

    cv = StratifiedKFold(n_splits = 10, shuffle=True)
    modelo = DecisionTreeClassifier(max_depth=2)
    results = cross_validate(modelo, x_azar, y_azar, cv = cv, return_train_score=False)
    imprime_resultados(results)
```

```
Accuracy médio: 75.78
Accuracy intervalo: [72.94, 78.62]
```

Gerando dados aleatórios de modelo de carro para simulação de agrupamento ao usar nosso estimador

```
[ ] np.random.seed(SEED)
    dados['modelo'] = dados.idade_do_modelo + np.random.randint(-2, 3, size=10000)
    dados.modelo = dados.modelo + abs(dados.modelo.min()) + 1
    dados.head()
```

	preco	vendido	idade_do_modelo	km_por_ano	modelo
0	30941.02	1	18	35085.22134	18
1	40557.96	1	20	12622.05362	24
2	89627.50	0	12	11440.79806	14
3	95276.14	0	3	43167.32682	6
4	117384.68	1	4	12770.11290	5

```
[ ] dados.modelo.unique()
```

```
array([18, 24, 14,  6,  5, 13, 20, 19, 15,  2, 17, 12, 11, 16,  3,  7, 21,
       23, 10,  9, 22,  8,  4,  1])
```

```
[ ] dados.modelo.value_counts()
```

```
20    901
19    798
18    771
21    723
17    709
16    668
14    621
22    575
15    573
13    557
12    511
11    401
10    371
23    370
9     336
8     278
7     206
24    199
6     181
5     108
4      76
3      44
2      17
1       6
Name: modelo, dtype: int64
```

Testando validação cruzada com GroupKFold

```
[ ] from sklearn.model_selection import GroupKFold

    SEED = 301
    np.random.seed(SEED)

    cv = GroupKFold(n_splits = 10)
    modelo = DecisionTreeClassifier(max_depth=2)
    results = cross_validate(modelo, x_azar, y_azar, cv = cv, groups = dados.modelo, return_train_score=False)
    imprime_resultados(results)
```

```
Accuracy médio: 75.78
Accuracy intervalo: [73.67, 77.90]
```

Cross validation com StandardScaler

```
[ ] from sklearn.preprocessing import StandardScaler
    from sklearn.svm import SVC

    SEED = 301
    np.random.seed(SEED)

    scaler = StandardScaler()
    scaler.fit(treino_x)
    treino_x_escalado = scaler.transform(treino_x)
    teste_x_escalado = scaler.transform(teste_x)

    modelo = SVC()
    modelo.fit(treino_x_escalado, treino_y)
    previsoes = modelo.predict(teste_x_escalado)

    acuracia = accuracy_score(teste_y, previsoes) * 100
    print("A acurácia foi %.2f%%" % acuracia)
```

```
A acurácia foi 74.40%
```

```
[ ] from sklearn.model_selection import GroupKFold

    SEED = 301
    np.random.seed(SEED)

    cv = GroupKFold(n_splits = 10)
    modelo = SVC()
    results = cross_validate(modelo, x_azar, y_azar, cv = cv, groups = dados.modelo, return_train_score=False)
    imprime_resultados(results)
```

```
Accuracy médio: 58.00
Accuracy intervalo: [56.10, 59.89]
```

```
[ ] scaler = StandardScaler()
    scaler.fit(x_azar)
    x_azar_escalado = scaler.transform(x_azar)
```

```
[ ] from sklearn.model_selection import GroupKFold

    SEED = 301
    np.random.seed(SEED)

    cv = GroupKFold(n_splits = 10)
    modelo = SVC()
    results = cross_validate(modelo, x_azar_escalado, y_azar, cv = cv, groups = dados.modelo, return_train_score=False)
    imprime_resultados(results)
```

```
Accuracy médio: 76.71
Accuracy intervalo: [74.30, 79.12]
```

```
from sklearn.pipeline import Pipeline

SEED = 301
np.random.seed(SEED)

scaler = StandardScaler()
modelo = SVC()

pipeline = Pipeline([('transformacao',scaler), ('estimador',modelo)])

cv = GroupKFold(n_splits = 10)
results = cross_validate(pipeline, x_azar, y_azar, cv = cv, groups = dados.modelo, return_train_score=False)
imprime_resultados(results)
```

```
Accuracy médio: 76.68
Accuracy intervalo: [74.28, 79.08]
```

```
[ ]
```


Produtos pagos do Colab - Cancelar contratos