INSTRUÇÕES

Instruções:

O dataset utilizado nesse assignment é proveniente do repositório de Machine Learning da UC Irvine. Trata-se de um dataset de vinhos amplamente utilizado em projetos de machine learning, e aqui focaremos no processo de EDA e modelagem como um todo.

Primeiramente, nesse assignmente iremos utilizar a biblioteca imbalanced-learn. Instale-a em seu ambiente ou no Colab antes de inicar suas atividades (*pip install imbalanced-learn*).

A seguir, certifique-se de seguir as etapas destas intruções para que seus resultados sejam similares aos do "gabarito" deste assignment, e consequentemente as mesmas respostas.

1) Una/concatene os datasets de vinho vermelho e branco (incluindo uma coluna informando qual o tipo de vinho, ou seja, vermelho ou branco).

2) Crie uma coluna *wine_quality* que será nosso target. Ela deve ser 1 caso o valor da qualidade (*quality*) do vinho seja superior a 6 (vinho "bom") ou 0 do contrário (vinho não classificado como "bom", poderíamos entender como mediano ou ruim).

3) Remova todas linhas duplicadas.

Dica para questão 4:

> Para responder a questão 4 será necessário padronizar as variáveis usando *StandardScaler()* ou scale() do sklearn. Essa é uma técnica muito útil para comparar a distribuição de variáveis em diferentes escalas, sobretudo para comparar outliers. Contudo, não **sobrescreva seus dados**, crie uma nova variável com estes valores padronizados. Através de um boxplot é fácil observar os ouliers e compará-los.

Dica para questão 5:

> Para responder a questão 5 faça a mesma padronização sugerida para questão 4, e quando for plotar o boxplot, utilize seaborn com o parâmetro `showfliers=False` para não mostrar os outliers (ignora os outliers).

4) Para reproduzir o pipeline do assignment:
   - Exclua a coluna *quality* das suas features, pois ela contém informação da target.
   - Separa os dados de treino e teste utilizando 30% para teste, e ***random_state=0***.
   - Faça um pré-processador para variáveis numéricas (*StandardScaler*) e categóricas (*OneHotEncoder*) e coloque-o em um pipeline que irá executar em seguida o modelo de regressão logística com auto-ajuste de regularização (C,

chamado *LogisticRegresionCV* no sklearn), com *random_state=0*. Segue print exemplo:

```
# Select numeric and categorical features
numerical_features = selector(dtype_include=np.number)
categorical_features = selector(dtype_exclude=np.number)
# Preprocessor
preprocessor = make_column_transformer((StandardScaler(), numerical_features), (OneHotEncoder(), categorical_features))
# Pipeline with logistic regression model
model_pipeline = Pipeline(steps=[('preprocessor', preprocessor), ('model', LogisticRegressionCV(random_state=0))])
# Train model
model_pipeline.fit(X_train, y_train)
```

- Responda a pergunta 6.
- Faça outro pipeline, desta vez adotando uma estratégia para minimizar a influência do desbalancemento da variável-alvo dataset. Uma vez que possivelmente não estejam familiarizados com a biblioteca imbalanced-learn, segue o código para esse pipeline, com os respectivos imports:

```
from imblearn.over_sampling import RandomOverSampler
from imblearn.pipeline import Pipeline as ImbPipeline

# Estratégia simples de oversampling (replicando linhas das variáveis com a classe minoritária)
oversampling = RandomOverSampler(sampling_strategy=.7, random_state=0)

# Preprocessador idêntico ao anterior
preprocessor = make_column_transformer((StandardScaler(), numerical_features), (OneHotEncoder(), categorical_features))

# Pipeline do pacote imbalanced-learn
model_imb_pipeline = ImbPipeline(steps=[
    ('oversampling', oversampling),
    ('preprocessor', preprocessor),
    ('model', LogisticRegressionCV(random_state=0))
])

# Train model
model_imb_pipeline.fit(X_train, y_train);
```

- Responda a pergunta 7 (predição do pipeline funciona do mesmo modo que do sklearn)

5) Utilize o pipeline do imblearn (segundo pipeline) com GridSearch do sklearn com 5 k-folds para testar diferentes modelos e verificar qual retorna maior F1 score. Teste os seguintes modelos:

- LogisticRegressionCV com *random_state=0*
- RandomForestClassifier com *random_state=0*
- GradientBoostingClassifier com *random_state=0*
- SVC com *random_state=0*

Lembre-se com pipeline no Grid search, é possível testar estes modelos de uma vez. Como o nome da step da pipeline que define o modelo é *model*, podemos passar estes parâmetros no grid search:

```
params = {
    'model': [
        LogisticRegressionCV(random_state=0),
        RandomForestClassifier(random_state=0),
        GradientBoostingClassifier(random_state=0),
        SVC(random_state=0)
    ]
}
```

6) Utilize gridsearch com o pipeline do imblearn para encontrar os melhores parâmetros do modelo SVC. Utilize a métrica F1 score e 5 k-folds. Utilize os seguintes parâmetros:

```
params = {
    'model__C': [0.01,0.1,1,10],
    'model__kernel': ['linear', 'poly', 'rbf', 'sigmoid', 'precomputed']
}
```

OBJOBJOBJ

Obs: As etapas para reproduzir o gabarito deste assignment não sifgnifica que sejam as únicas possíveis, tampouco as melhores. No entanto, o cumprimento das etapas é importante para garantir a correção apropriada do case.