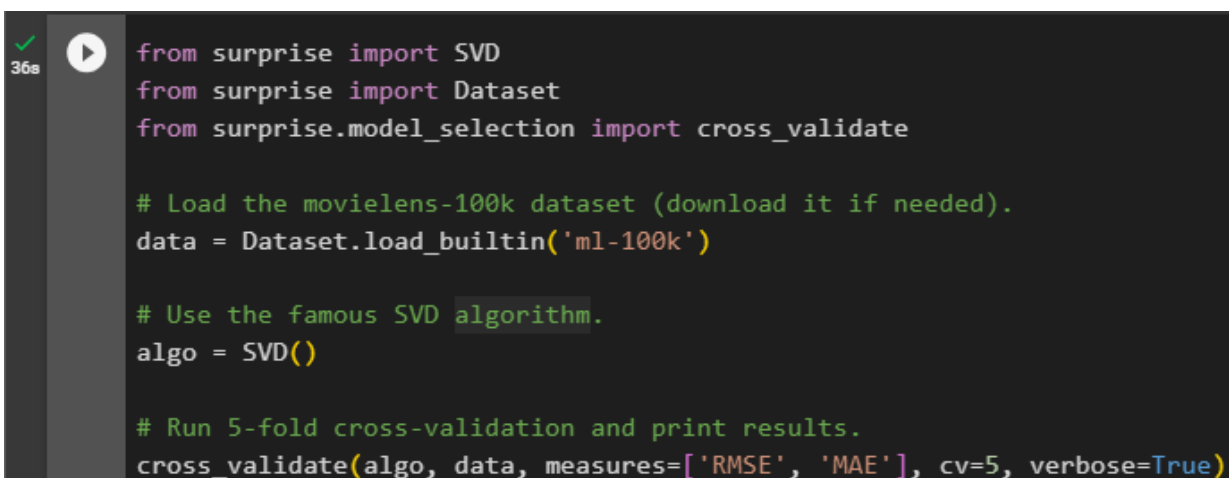


## Trabalho 4 – Sistemas de Recomendação

**Surprise** é uma biblioteca Python que serve para construir e analisar sistemas de recomendação, e já possui alguns datasets inclusos, como o do Movielens. **Cross-validation** é uma técnica estatística usada para avaliar o desempenho de um modelo de machine learning. Ela é ideal para quando se deseja estimar a capacidade de generalização de novos dados em datasets limitados, porque a sua ideia básica é dividir o dataset em partes (folds), treinar o modelo em algumas dessas partes e testá-lo nas partes restantes. Isso é repetido várias vezes, garantindo que cada parte do conjunto de dados seja usada tanto para treinamento quanto para teste. Assim, o cross-validation fornece uma estimativa mais robusta do desempenho do modelo do que se fosse utilizada apenas uma única divisão de treino e teste.

No algoritmo a seguir, na linha “data = Dataset.load\_builtin('ml-100k')” está sendo carregado para a variável “data” o dataset do Movielens com cem mil registros. Na linha “algo = SVD()” está sendo criada uma instância do algoritmo que será utilizado na validação cruzada, no caso o SVD (que basicamente decompõe uma matriz em três outras matrizes componentes). Já na linha “cross\_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)” a cross-validation está sendo realizada, passando como parâmetro o dataset, o algoritmo, as duas métricas que devem ser calculadas (RMSE e MAE), o número de folds em que ele deve ser dividido e que informações detalhadas devem ser exibidas nos resultados (verbose=true).

A screenshot of a Jupyter Notebook cell. On the left, there is a green checkmark icon and a play button icon, with the text '36s' below them. The main area contains Python code for using the Surprise library. The code imports SVD, Dataset, and cross\_validate from surprise. It then loads the 'ml-100k' dataset, creates an SVD algorithm instance, and runs a 5-fold cross-validation with RMSE and MAE as measures, displaying verbose output.

```
from surprise import SVD
from surprise import Dataset
from surprise.model_selection import cross_validate

# Load the movielens-100k dataset (download it if needed).
data = Dataset.load_builtin('ml-100k')

# Use the famous SVD algorithm.
algo = SVD()

# Run 5-fold cross-validation and print results.
cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

O seguinte resultado é exibido no console após a execução do código acima:

```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).
```

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.9259	0.9336	0.9434	0.9402	0.9413	0.9369	0.0064
MAE (testset)	0.7312	0.7349	0.7415	0.7399	0.7430	0.7381	0.0044
Fit time	1.75	2.19	1.41	1.39	1.39	1.62	0.31
Test time	0.22	0.31	0.12	0.21	0.20	0.21	0.06

As métricas **RMSE** e **MAE** servem para avaliar o quão bem um modelo está realizando previsões. Valores mais baixos indicam um melhor desempenho. Durante a cross-validation, essas métricas são calculadas para cada fold e, ao final, as médias (Mean) são geralmente apresentadas como uma indicação do desempenho geral do modelo.

A métrica RMSE é calculada pela raiz quadrada da média dos quadrados dos erros, onde o erro bruto é a diferença entre o valor previsto pelo modelo e o valor real. A RMSE penaliza erros extremos mais do que erros pequenos. Já a MAE é uma medida da média das diferenças absolutas entre os valores previstos e os valores reais, sendo menos sensível a erros extremos do que a RMSE.

Analisando os resultados gerados, é possível observar que as duas métricas foram calculadas para cada fold do dataset, a MAE tendo tempos de análise e teste (Fit time e Test time) significativamente menores do que a RMSE, devido à maior simplicidade da sua fórmula. Como a RMSE penaliza mais erros extremos, ela resultou em valores maiores do que a MAE, indicando um menor desempenho ao modelo. Além da média dos valores das métricas (Mean) também é calculado o seu desvio padrão (Std).

É possível também utilizar a função `get_neighbors()` da biblioteca Surprise para obter registros similares a outro em um dataset, quando se está utilizando algoritmos que utilizam uma medida de similaridade como o KNN. No exemplo a seguir são retornados os 10 filmes mais próximos do filme Toy Story no dataset do MovieLens:

```

import io # noqa

from surprise import Dataset, get_dataset_dir, KNNBaseline

def read_item_names():
    """Read the u.item file from MovieLens 100-k dataset and return two
    mappings to convert raw ids into movie names and movie names into raw ids.
    """

    file_name = get_dataset_dir() + "/ml-100k/ml-100k/u.item"
    rid_to_name = {}
    name_to_rid = {}
    with open(file_name, encoding="ISO-8859-1") as f:
        for line in f:
            line = line.split("|")
            rid_to_name[line[0]] = line[1]
            name_to_rid[line[1]] = line[0]

    return rid_to_name, name_to_rid

# First, train the algorithm to compute the similarities between items
data = Dataset.load_builtin("ml-100k")
trainset = data.build_full_trainset()
sim_options = {"name": "pearson_baseline", "user_based": False}
algo = KNNBaseline(sim_options=sim_options)
algo.fit(trainset)

# Read the mappings raw id <-> movie name
rid_to_name, name_to_rid = read_item_names()

# Retrieve inner id of the movie Toy Story
toy_story_raw_id = name_to_rid["Toy Story (1995)"]
toy_story_inner_id = algo.trainset.to_inner_iid(toy_story_raw_id)

# Retrieve inner ids of the nearest neighbors of Toy Story.
toy_story_neighbors = algo.get_neighbors(toy_story_inner_id, k=10)

# Convert inner ids of the neighbors into names.
toy_story_neighbors = (
    algo.trainset.to_raw_iid(inner_id) for inner_id in toy_story_neighbors
)
toy_story_neighbors = (rid_to_name[rid] for rid in toy_story_neighbors)

print()
print("The 10 nearest neighbors of Toy Story are:")
for movie in toy_story_neighbors:
    print(movie)

```

Na função `read_item_names`, o arquivo `u.item` do dataset é lido e dois mapeamentos são feitos: `rid_to_name` mapeia identificadores brutos (raw ids) para nomes

de filmes e `name_to_rid` mapeia nomes de filmes para identificadores brutos (raw ids são identificadores originais retirados do dataset antes do processamento). Após o mapeamento é realizado o treinamento do algoritmo. Na linha `Dataset.load_builtin("ml-100k")` o dataset do Movielens é carregado, e na linha `build_full_trainset()` um conjunto de treinamento completo é construído a partir dos dados. Na linha `sim_options = {"name": "pearson_baseline", "user_based": False}` as opções de similaridade são configuradas especificando que a métrica de similaridade deve ser "pearson\_baseline" e que o método é baseado em itens ("user\_based": False). Por fim, o algoritmo KNNBaseline é inicializado com essas opções e treinado com `algo.fit(trainset)`.

Na linha `rid_to_name, name_to_rid = read_item_names()`, a função `read_item_names` é usada para obter os mapeamentos `rid_to_name` e `name_to_rid`. Na seção `# Retrieve inner id of the movie Toy Story`, o código converte o nome do filme "Toy Story (1995)" em seu identificador bruto (`toy_story_raw_id`) e, em seguida, encontra o identificador interno correspondente no conjunto de treinamento (`toy_story_inner_id`).

Na linha `toy_story_neighbors = algo.get_neighbors(toy_story_inner_id, k=10)` os 10 vizinhos mais próximos de `toy_story_inner_id` são obtidos. Após isso, os identificadores internos dos vizinhos retornados são convertidos de volta para seus nomes nas seguintes linhas: `toy_story_neighbors = (algo.trainset.to_raw_iid(inner_id) for inner_id in toy_story_neighbors)` e `toy_story_neighbors = (rid_to_name[rid] for rid in toy_story_neighbors)`.

Por fim, os nomes são impressos no resultado:

```
Estimating biases using als...
Computing the pearson_baseline similarity matrix...
Done computing similarity matrix.

The 10 nearest neighbors of Toy Story are:
Beauty and the Beast (1991)
Raiders of the Lost Ark (1981)
That Thing You Do! (1996)
Lion King, The (1994)
Craft, The (1996)
Liar Liar (1997)
Aladdin (1992)
Cool Hand Luke (1967)
Winnie the Pooh and the Blustery Day (1968)
Indiana Jones and the Last Crusade (1989)
```