Nhập môn KHDL - Đồ án cuối kỳ

Đánh giá – Gợi ý Laptop đáng mua nhất trong top 100 sản phẩm TiKi

Nhóm 43: - 1412647 Nguyễn Ngọc Vũ
- 1712804 Võ Minh Thư

Phần 1: Thu thập dữ liệu

- Bước đầu tiên là thi thập link top 100 sản phẩm Laptop từ Tiki.
- Sau đó tiền hành thập dữ liệu bằng cách parse HTML

```
from urllib.request import urlopen
import requests
from bs4 import BeautifulSoup
import json
import time
import pandas as pd
import datetime as dt
import re
f = open("tiki.txt", "r")
lines = []
for line in f:
    lines.append(line.strip('\n'))

file = open('products.csv', 'w', encoding='utf-8')
file.write('brand\tcpu\tram\tssd\tmonitor\tprice\trating_value\n')
```

- Các thư viện được dùng như trong ảnh.
- File tiki là file dùng để lưu các liên kết sản phẩm, data crawl gồm các thông tin: brand, cpu, ram, minitor, price, rating và được ghi vào file products.csv

```
headers = {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_5) AppleWebKit/537.36 (KHTML, like Gecko) Chr

html_text = requests.get(url, headers=headers).text

soup = BeautifulSoup(html_text, "html.parser")

# Get Title
title = soup.find('h1', attrs={'class':'title'}).string

# Get brand
brand = str(soup.find('meta', attrs={'name':'brand'})['content']).strip()

# Extract CPU, RAM, SSD, Monitor from Title
## replace Ryzen 3, 5, 7 to R3, R5, R7
title_str = str(title).replace('Ryzen 3', 'R3').replace('Ryzen 5', 'R5').replace('Ryzen 7', 'R7')
## extract cpu
cpu = re.search(r'M1|(i3|i5|i7|r3|r5|r7).\w+|Pentium.\w+.\d+\w+', title_str, re.IGNORECASE)
cpu = cpu.group(0).replace(' ', '-') if cpu is not None else ''
## extract ram
ram = re.search(r'\d+(GB)', title_str, re.IGNORECASE)
end_ram = ram.end() if ram is not None else 0
ram = re.search(r'\d+', ram.group(0)).group(0) if ram is not None else ''
## extract ssd
ssd = re.search(r'\d+(GB)', title_str[end_ram:], re.IGNORECASE)
ssd = re.search(r'\d+', ssd.group(0)).group(0) if ssd is not None else ''
## extract monitor
monitor = re.search(r'\d+(.\d+|)(|.|..)((|\w|Full )HD|[iI]nch)', title_str, re.IGNORECASE)
monitor = re.search(r'\d+(.\d+|)', monitor.group(0)).group(0) if monitor is not None else ''

# Get Price
price= soup.find('div', attrs={"class":"left", "itemprop":"offers"}).find('meta', attrs={'itemprop': 'price'})['cont

# Get rating
rating_valueStr = soup.find('div', attrs={'itemprop': 'aggregateRating'})
if rating_valueStr:
    rating_value = rating_valueStr.find('meta', attrs={'itemprop': 'ratingValue'})['content']
else:
    rating_value = 0
```

Phần 2: Khám phá dữ liệu

- Ở bước này chúng ta khám phá về mô hình dữ liệu: bao nhiêu dòng, bao nhiêu cột, loại dữ liệu cuả data.

Out[3]:

	brand	cpu	ram	ssd	monitor	price	rating_value
0	HP	i5-1035G4	8.0	512.0	13.3	22890000	4.5
1	HP	i7-10750H	8.0	512.0	15.6	25990000	5.0
2	HP	i5-8265U	8.0	512.0	13.3	39990000	0.0
3	HP	i7-10750H	8.0	512.0	15.6	27139000	0.0
4	Apple	i5-8th	8.0	256.0	13	30498000	4.5

Dữ liệu có bao nhiêu dòng và bao nhiêu cột?

```
In [4]: data_df.shape
Out[4]: (100, 7)
```

Kiểu dữ liệu của data

```
In [5]: data_df.dtypes
Out[5]: brand           object
        cpu             object
        ram            float64
        ssd            float64
        monitor         object
        price            int64
        rating_value   float64
        dtype: object
```

Từ đó đặt ra câu hỏi
- Output - *rating_value* được tính từ Input - *các thông tin của laptop* như thế nào?
- Việc tìm ra công thức này giúp ta có thể dự đoán được sự yêu thích của các laptop mới được sản xuất để xem có đáng mua hay không?

Phần 3: Khám phá dữ liệu cột output

- Cột Output có kiểu dữ liệu gì?

```
In [6]: data_df['rating_value'].dtype
```

```
Out[6]: dtype('float64')
```

- Cột output có giá trị thiếu không?

```
In [7]: data_df['rating_value'].isna().sum()
```

```
Out[7]: 0
```

- Tỉ lệ các lớp trong cột output?

```
In [8]: data_df['rating_value'].value_counts(normalize=True) * 100
```

```
Out[8]: 0.0    32.0
        5.0    30.0
        4.5    27.0
        4.0     7.0
        3.5     3.0
        4.2     1.0
        Name: rating_value, dtype: float64
```

Phần 4: Tiền xử lý (tách các tập)

4. Tiền xử lý (tách các tập)

- Tách tập x, y

In [9]:
```
y_sr = data_df['rating_value'].map(lambda x: 0 if x < 3 else 1)
x_df = data_df.drop("rating_value", axis=1)
```

- Tách tập huấn luyện và tập validation theo tỉ lệ 70%:30%

In [10]:
```
train_x_df, val_x_df, train_y_sr, val_y_sr = train_test_split(x_df, y_sr, test_size=0.3, stratify=y_sr, random_state=0)
```

In [11]:
```
train_x_df.head().index
```

Out[11]:
```
Int64Index([84, 67, 69, 8, 62], dtype='int64')
```

Phần 5: Khám phá dữ liệu (tập huấn luyện)

- Phần này ta khám phá cấu trúc, loại dữ liệu, phân bố của các tập huấn luyện.

Mỗi cột input hiện đang có kiểu dữ liệu gì? Có cột nào có kiểu dữ liệu chưa phù hợp để có thể xử lý tiếp không?

In [12]:
```
train_x_df.dtypes
```

Out[12]:
```
brand       object
cpu         object
ram        float64
ssd        float64
monitor     object
price        int64
dtype: object
```

- Ở đây cột monitor, ssd đang có kiểu là object trong khi nó nên phải là float.

> Mình chuyển cột monitor thành float

In [13]:
```
train_x_df.dtypes
```

Out[13]:
```
brand       object
cpu         object
ram        float64
ssd        float64
monitor     object
price        int64
dtype: object
```

- Với mỗi cột input có kiểu dữ liệu dạng số, các giá trị được phân bố như thế nào?

Phân bố của input dạng số và không phải dạng số

Với mỗi cột input có kiểu dữ liệu dạng số, các giá trị được phân bố như thế n

Trong train_x_df có 5/7 cột có kiểu dữ liệu dạng số

In [14]:
```
train_x_df.dtypes[train_x_df.dtypes != object]
```

Out[14]:
```
ram      float64
ssd      float64
price      int64
dtype: object
```

In [15]:
```
num_cols = ['ram', 'ssd', 'monitor', 'price']
df = train_x_df[num_cols]
def missing_ratio(df):
    return (df.isna().mean() * 100).round(1)
def lower_quartile(df):
    return df.quantile(0.25).round(1)
def median(df):
    return df.quantile(0.5).round(1)
def upper_quartile(df):
    return df.quantile(0.75).round(1)
df.agg([missing_ratio, 'min', lower_quartile, median, upper_quartile, 'max'])
```

Out[15]:

	ram	ssd	monitor	price
missing_ratio	12.9	15.7	7.1	0.0
min	4.0	2.0	NaN	8288000.0
lower_quartile	4.0	256.0	NaN	13179000.0
median	8.0	256.0	NaN	17109500.0
upper_quartile	8.0	512.0	NaN	23261750.0
max	16.0	512.0	NaN	48999000.0

Với mỗi cột input có kiểu dữ liệu không phải dạng số, các giá trị được phân bố như thế nào?

In [16]:
```
pd.set_option('display.max_colwidth', 200) # Để nhìn rõ hơn
cat_cols = list(set(train_x_df.columns) - set(num_cols))
df = train_x_df[cat_cols]
def missing_ratio(df):
    return (df.isna().mean() * 100).round(1)
def num_values(df):
    return df.nunique()
def value_ratios(c):
    return dict((c.value_counts(normalize=True) * 100).round(1))
df.agg([missing_ratio, num_values, value_ratios])
```

Out[16]:

	brand	cpu
missing_ratio	0.0	7.1
num_values	9	32
value_ratios	{'Asus': 32.9, 'Dell': 17.1, 'Apple': 14.3, 'HP': 11.4, 'Lenovo': 8.6, 'MSI': 7.1, 'Acer': 5.7, 'Microsoft': 1.4, 'LG': 1.4}	{'i5-1035G1': 13.8, 'i5-1135G7': 6.2, 'M1': 6.2, 'i3-10110U': 6.2, 'i5-10210U': 4.6, 'i7-10750H': 4.6, 'R7-4800H': 4.6, 'i5-10300H': 3.1, 'i3-1005G1': 3.1, 'i7-10510U': 3.1, 'R3-3200U': 3.1, 'i7-1...

Phần 6: Tiền xử lý (tập huấn luyện)¶

Tạo một class tên là *ColumnModifier* để chuyển các tập dữ liệu dạng số về dạng số

In [17]:
```
class ColumnModifier(BaseEstimator, TransformerMixin):
    def __init__(self, num_columns = []):
        self.num_columns = num_columns;
    def fit(self, X_df, y=None):
        return self
    def transform(self, X_df, y=None):
        result_df = X_df;
        for col in self.num_columns:
            result_df.loc[:, col] = pd.to_numeric(result_df[col], errors='coerce')
        return result_df;
```

In [18]:
```
pre = ColumnModifier(['ram', 'ssd', 'price', 'monitor']);
pre.fit_transform(train_x_df).dtypes
```

```
/home/nnvuf/.local/lib/python3.8/site-packages/pandas/core/indexing.py:1675: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  self._setitem_single_column(ilocs[0], value, pi)
```

Out[18]:
```
brand       object
cpu         object
ram        float64
ssd        float64
monitor    float64
price        int64
dtype: object
```

Tạo một pipeline để xử lý các việc sau:

- Với các cột dạng số, ta sẽ điền giá trị thiếu bằng giá trị mean của cột. Với tất cả các cột dạng số trong tập huấn luyện, ta đều cần tính mean, vì ta không biết được cột nào sẽ bị thiếu giá trị khi dự đoán với các véc-tơ input mới.
- Với các cột không phải dạng số và không có thứ tự:
  - Ta sẽ điền giá trị thiếu bằng giá trị mode (giá trị xuất hiện nhiều nhất) của cột. Với tất cả các cột không có dạng số và không có thứ tự, ta đều cần tính mode, vì ta không biết được cột nào sẽ bị thiếu giá trị khi dự đoán với các véc-tơ input mới.
  - Sau đó, ta sẽ chuyển sang dạng số bằng phương pháp mã hóa one-hot
- Với cột không phải dạng số và có thứ tự (cột "monitor"):
  - Ta sẽ điền giá trị thiếu bằng giá trị mode (giá trị xuất hiện nhiều nhất) của cột.
- Cuối cùng, khi tất cả các cột đã được điền giá trị thiếu và đã có dạng số, ta sẽ tiến hành chuẩn hóa bằng cách trừ đi mean và chia cho độ lệch chuẩn của cột để giúp cho các thuật toán cực tiểu hóa như Gradient Descent, LBFGS, ... hội tụ nhanh hơn

In [19]:
```
num_cols = ['ram', 'ssd', 'price', 'monitor']
unorder_cate_cols = ['brand', 'cpu']
preprocess_pipeline = Pipeline([
    ('column_modifier', ColumnModifier(num_cols)),
    ('transformer', ColumnTransformer([
        ('num_transformer', SimpleImputer(missing_values=np.nan, strategy="mean"), num_cols),
        ('unorder_transformer', make_pipeline(
            SimpleImputer(strategy="most_frequent"),
            OneHotEncoder(handle_unknown='ignore')), unorder_cate_cols),
#         ('order_cate_cols', SimpleImputer(strategy="most_frequent"), order_cate_cols)
    ], remainder='passthrough')),
    ('std_scaler', StandardScaler(with_mean=False))
])

preprocessed_train_X = preprocess_pipeline.fit_transform(train_x_df);
```

In [20]:
```
preprocess_pipeline
```

Phần 7: Tiền xử lý (tập validation

```
In [21]: preprocessed_val_X = preprocess_pipeline.transform(val_x_df)

/home/nnvuf/.local/lib/python3.8/site-packages/pandas/core/indexing.py:1675: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  self._setitem_single_column(ilocs[0], value, pi)
```

Một khi đã có *preprocess_pipeline* với các giá trị đã được tính từ tập huấn luyện, ta có thể dễ dàng dùng phương thức *transform* để tiền xử lý cho các véc-tơ input mới trong tập validation và tập kiểm tra

Phần 8: Tiền xử lý + mô hình hóa

Tạo full pipeline với những công việc như ở bước 6 và áp dụng mô hình Neural Net để phân lớp

In [22]:
```
num_cols = ['ram', 'ssd', 'price', 'monitor']
unorder_cate_cols = ['brand', 'cpu']
full_pipeline = Pipeline([
    ('column_modifier', ColumnModifier(num_cols)),
    ('transformer', ColumnTransformer([
        ('num_transformer', SimpleImputer(missing_values=np.nan, strategy="mean"), num_cols),
        ('unorder_transformer', make_pipeline(
            SimpleImputer(strategy="most_frequent"),
            OneHotEncoder(handle_unknown='ignore')), unorder_cate_cols),
#         ('order_cate_cols', SimpleImputer(strategy="most_frequent"), order_cate_cols)
    ], remainder='passthrough')),
    ('std_scaler', StandardScaler(with_mean=False)),
    ('modeling', MLPClassifier(
        hidden_layer_sizes=(20, ),
        activation='tanh',
        solver='lbfgs',
        random_state=0,
        max_iter=2500,
        alpha=0.1
    ))
])
```

- Thử nghiệm với các giá trị khác nhau của các siêu tham số
- và chọn ra các giá trị tốt nhất

In [23]:
```
train_errs = []
val_errs = []
alphas = [0.1, 1, 10, 100, 1000]
best_val_err = float('inf'); best_alpha = None
for alpha in alphas:
    full_pipeline.set_params(modeling__alpha=alpha);
    full_pipeline.fit(train_x_df, train_y_sr);
    val_y_pred = full_pipeline.predict(val_x_df);
    train_err = (1 - full_pipeline.score(train_x_df, train_y_sr)) * 100;
    val_err = (1 - full_pipeline.score(val_x_df, val_y_sr)) * 100;
    train_errs.append(train_err);
    val_errs.append(val_err);
    if val_err < best_val_err:
        best_val_err = val_err;
        best_alpha = alpha;
```

Phần 9: Huấn luyện lại mô hình

9. Huấn luyện lại mô hình với tập x_df và y_sr

In [25]:
```
full_pipeline.set_params(modeling__alpha=best_alpha);
full_pipeline.fit(x_df, y_sr);
```

Cuối cùng, bạn sẽ huấn luyện lại full_pipeline trên x_df và y_sr (tập huấn luyện + tập validation) với *best_alpha* tìm được ở trên để ra được mô hình cụ thể cuối cùng.

Phần 10: Nhìn lại quá trình làm đồ án

10.1. Khó khăn:

- Không có

10.2. Hữu ích:

- Hiểu được thêm về quy trình khoa học dữ liệu
  - Hiểu được cách crawl data như thế nào
  - Hiểu được một pipeline hoạt động ra sao

10.3. Nếu có thêm thời gian:

- Nếu có thêm thời gian thì có thể tìm hiểu được thêm nhiều data hơn. do độ chính xác vẫn còn thấp