ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

MÔN HỌC: TOÁN ỨNG DỤNG VÀ THỐNG KÊ

BÁO CÁO ĐỒ ÁN CUỐI KÌ

HỒI QUY TUYẾN TÍNH – HỒI QUY LOGISTIC

Giáo viên hướng dẫn: Vũ Quốc Hoàng - Trần Thị Thảo Nhi

I. THÔNG TIN NHÓM VÀ PHÂN CÔNG

Họ tên	MSSV	Phân công	Đánh giá
Trịnh Vũ Minh Hùng	1712049	- Quản lý tiến độ làm việc của các thành viên. - Tìm hiểu mô hình Linear Regression, đưa ra giải pháp chuẩn hóa tập dữ liệu. - Tổng hợp, và viết báo cáo.	100%
Lê Hoài Bảo	1712005	- Viết chương trình tham số dòng lệnh (cmd) cho mô hình Linear Regression - Hỗ trợ tìm hiểu mô hình Linear regression	100%
Nguyễn Văn Khoa	1712072	- Cài đặt mô hình Linear Regression. - Viết báo cáo.	100%
Mai Công Trình	1712840	- Cài đặt mô hình Logistic Regression, đưa ra kỹ thuật để giải quyết vấn đề của tập dữ liệu. - Viết báo cáo.	100%
Trần Đình Khoát	1712073	- Hỗ trợ tìm hiểu mô hình Logistic Regression, Linear Regression. - Đưa ra các giải pháp chuẩn hóa tập dữ liệu thô ban đầu.	100%

II. ĐỒ ÁN 1 – HỒI QUY TUYẾN TÍNH (LINEAR REGRESSION)

1. Các thư viện cần thiết

```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

sns.set()
```

2. Các hàm cần thiết

- **Clear function:** Hàm loại bỏ các trường dữ liệu không ảnh hướng đến giá xe (price)

```
def clear(df):
df = data.drop(['odometer'],axis =1 )
df = df .drop(['engineType'],axis =1 )
df = df .drop(['engineCapacity'],axis =1 )
df = df .drop(['photos'],axis =1 )
return df
```

- **Handle outlier**: Loại bỏ những giá trị nằm cách xa khoảng trung bình

```
# manufacturer
plt.rcParams['figure.figsize'] = (22, 10)
df1 = data_cleared.copy()
```

```
a = df1['manufacturer'].value_counts()
#sns.distplot(a.values)
q= a.quantile(0.99)
b=a[a<q]
#sns.distplot(b.values)
df2 = df1[df1['manufacturer']!='Volkswagen']
df2
```

```
#color
a = df2['color'].value_counts()
a
sns.distplot(a.values)
q= a.quantile(0.99)
b=a[a<q]
b
sns.distplot(b.values)
df3 = df2[df2['color']!='black']
df3
```

```
#bodyType
a = df3['bodyType'].value_counts()
a
sns.distplot(a.values)
q= a.quantile(0.99)
b=a[a<q]
sns.distplot(b.values)
df4 = df3[df3['bodyType']!='sedan']
df4
```

```
# year
q = df4['year'].quantile(0.01)
data = df4[df4['year'] >q]
data['year'].describe()
data
sns.distplot(data['year'])
```

- Lấy các giá trị dummies

```
data_dummies = pd.get_dummies(data, drop_first=True)
data_dummies= data_dummies.astype(float)
```

- Scale data về dạng chuẩn

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(inputs)
```

- Chạy thuật toán Linear Regression

```
reg = LinearRegression()
reg.fit(x_train,y_train)
## giá trị bias
reg.intercept_
## các biến x1, x2, ... ,xn
temp = pd.DataFrame(inputs.columns.values , columns = ['features'])
temp['weights'] = reg.coef_
temp
df_pf = pd.DataFrame(np.exp(y_hat_test), columns=['Prediction'])
```

3. Quy trình thực hiện

- Bước 1: Đọc dữ liệu từ file X_Train.csv , Y_train.csv
- Bước 2: Tiền xử lý (xóa các dòng có missing values)
- Bước 3: Trực quan các trường dữ liệu để tìm ra những trường dữ liệu ảnh hưởng đến giá xe
- Bước 4: Xóa bớt các trường không cần thiết
- Bước 5: Thực hiện bài toán Linear regression dựa theo những trường dữ liệu còn lại

4. Bảng kết quả đo độ chính xác của mô hình

	Prediction	Target
0	4064.272205	4500.0
1	7496.546620	7500.0
2	12024.805575	7800.0
3	12499.420914	12900.0
4	13917.527426	12700.0

	Prediction	Target	different	%
count	2844.000000	2844.000000	2844.000000	2844.000000
mean	6146.694659	6348.476442	201.781783	26.245736
std	5659.986102	5981.547325	2398.919923	31.408014
min	538.207434	260.000000	-13123.632521	0.002411
25%	2117.855271	2300.000000	-603.090409	8.888098
50%	4259.718395	4500.000000	80.806898	18.346767
75%	8232.081180	8628.250000	889.020297	32.332885
max	43980.687467	49950.000000	30982.054055	422.243731

⇨ **Nhận xét:** Từ cột %, ta thấy được rằng: mô hình dự đoán khá chính xác với các mức giá xe từ $low \rightarrow medium$. Còn những giá giá quá cao (high) thì độ sai lệch là rất lớn.

III. ĐỒ ÁN 2 – HỒI QUY LOGISTIC (LOGISTIC REGRESSION)

1. Các thư viện cần thiết

```
# các thu viện cần thiết
import pandas as pd
import numpy as np
import pickle
from matplotlib import pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

2. Các hàm cần thiết

- Load data:

```
def loadData(path):
'''
Thực hiện đọc dữ liệu từ file .csv

Input:
path: str
đường dẫn đến file .cvs cần đọc dữ liệu
Output:
data: ndarray, dtype=int64
chứa data của dữ liệu
label: ndrray, dtype=int64
chứa nhãn của dữ liệu
'''

# load data
df = pd.read_csv("train.csv")

# truy xuat data values
database = df.values
```

```
# truy xuất data
data = database[:,1:]
# truy xuất label
labels = database[:,0]

# trả về data, label
return data, labels
```

- Chuyển data thành dữ liệu ảnh

```
def data2Img(data):
'''
Thực hiện chuyển đổi cơ sở dữ liệu data ở dạng dòng thành data ma trận hì
nh ảnh

Input:
data: ndarray, dtype=int64
data chứa cơ sở dữ liệu dòng của hình ảnh
Output:
images: adarray, dtype=int8
chứa ma trận các hình ảnh đã được chuyển đổi
'''

# khi doc vao la kieu int64 nen ta can chuyen ve du lieu uint8 (la kieu du li
eu cua anh)
data_t = data.astype(np.uint8)

# convert ve ma tran 28x28 va luu vao list images
images = []

for _data in data_t:
_data = _data.reshape(-1,28)
images.append(_data)
# bien list thanh numpy.narray
images = np.array(images)
```

```
# trả về tập hình ảnh
return images
```

- Hiển thị hình ảnh theo label

```
def showImgByLabel(images, labels, label, n):
'''
Thực hiện show hình ảnh theo label được cung cấp

Input:
images: ndarray, dtype=int8
tập hình ảnh của cơ sở dữ liệu
labels: ndarray
tập nhãn của cơ sở dữ liệu
label: type=int
nhãn truyền vào muốn hiển hình ảnh (giới hạn 0 -> 24)
n: type=int
số lượng hình muốn hiển thị ở nhãn đó
Output:
out: hình ảnh được hiển thiện trên màn hình
'''

count = 0; # biến đếm tổng số hình
for i in range(images.shape[0]):
# kiểm tra label
if labels[i] == label:
# hiển thị hình ảnh lên màn hình
plt.imshow(images[i], cmap="gray")
plt.show()
count += 1

# kiểm tra đã đủ số hình chưa
if count == n:
break
```

- Lưu model

```
def saveModel(model, path):
'''
Lưu model đã được đào tạo

Input:
model:
model được đào tạo cần lưu
path: str
đường dẫn cần lưu
Output
out: model được lưu ra file
'''

pickle.dump(model, open(path,'wb'))
```

- Load model

```
def loadModel(path):
'''
Tải lên model từ file

Input:
path: str
Đường dẫn đến file chứa model
Output:
model: biến chứa model
'''

# load model
model = pickle.load(open(path,'rb'))

# trả về
return model
```

- Hàm train Scaler

```
def trainStandardScaler(data):
'''
Đào tạo mô hình

Chuẩn hóa các tính năng bằng cách loại bỏ giá trị trung bình và tỷ lệ theo phương sai đơn vị

Điểm chuẩn của một mẫu x được tính như sau:

z = (x - u) / s

Input:
data: ndarray
data làm chuẩn cần training
Output:
scaler: sklearn.preprocessing.StandardScaler
model scaler đã được đào tạo
'''

scaler = StandardScaler()
# Fit on training set only.
scaler.fit(data)

return scaler
```

- Train PCA

```
def trainPCA(data, n):
'''
Đào tạo mô hình PCA

Input:
data: ndarray
```

```
data cần đào tạo
n: int
số thành phần cần giữ lại
Output:
pca: sklearn.decomposition.PCA
model PCA đã được đào tạo
'''

# Make an instance of the Model
pca = PCA(n)

# train
pca.fit(data)

return pca
```

- Train Logistic Regression

```
def trainLogisticRegression(data, label):
'''
Đào tạo mô hình hồi quy Logistic

Input:
data: ndarray
Data cần đào tạo
label: ndarray
Nhãn tương ứng với data
Ouput:
model: sklearn.linear_model.LogisticRegression
model đã được đào tạo
'''

# all parameters not specified are set to their defaults
# default solver is incredibly slow which is why it was changed to 'lbfgs'
model = LogisticRegression(solver = 'lbfgs')
```

```
# train
model.fit(data, label)

return model
```

- Test model

```
def testLogisticRegression(data, label, model):
'''
Kiểm tra độ chính xác của mô hình hồi quy Logistic

Input:
data: ndarray
data chứa dữ liệu
label: ndarray
nhãn của dữ liệu
model: sklearn.linear_model.LogisticRegression
model của hồi quy Logistoc
Output:
out: phần trăm độ chính xác của mô hình
'''

# Predict for One Observation (image)
sum = 0
for i in range(data.shape[0]):
if model.predict(data[i].reshape(1,-1))==label[i]:
sum += 1

print('Accuracy of the model on the %d test images: %d %%' % (data.shap
e[0], 100 * sum / data.shape[0]))
```

- Tiền xử lí liệu

```
def preprocessingData(data, label):
```

```
'''
Tiền xử lí dữ liệu data và nhãn tương ứng:
data thì sử dụng 2 kĩ thuật (scaler, pca),
label thì chuyển đổi kiểu dữ liệu

Input:
data: ndarray
data cần tiền xử lí
label: ndarray
label cần tiền xử lí
Output:
data_new: ndarray
data đã được xử lí
label_new: ndarray
label đã được chuyển đổi kiểu dữ liệu
'''

# chuyển đổi kiểu dữ liệu cho data thành float64
data_new = data.astype(np.float64)
# chuyển labels thành int8
label_new = label.astype(np.int8)

# sử dụng kĩ thuật scaler

# load model scaler
scaler = loadModel('scaler.pkl')

# Apply
data_new = scaler.transform(data_new)

# sử dụng PCA

# load model PCA
pca = loadModel('pca.pkl')
```

```
# Apply
data_new = pca.transform(data_new)

return data_new, label_new
```

3. Quy trình thực hiện

- Bước 1: Đọc dữ liệu từ file
  Mô tả dữ liệu:
  - Dữ liệu gồm 27 000 mẫu
  - Ảnh trắng đen (grayscale), kích thước 28x28 pixel và được đánh nhãn có giá trị từ 0 đến 24

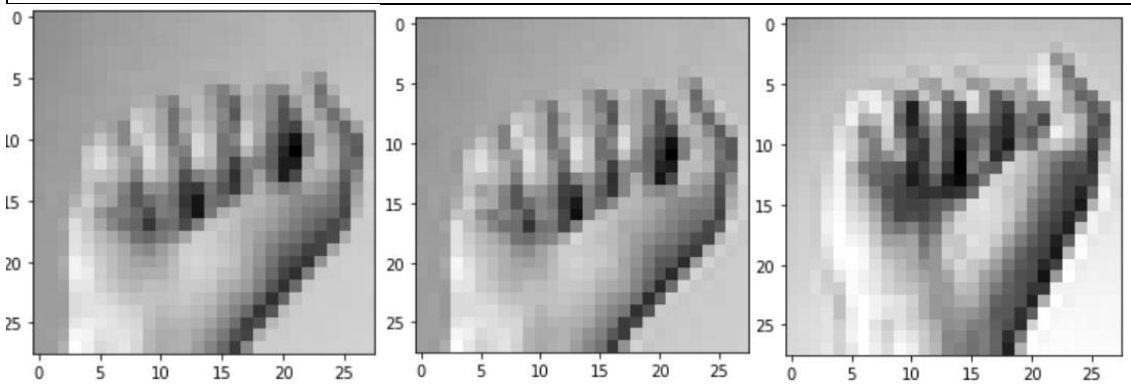
```
# load dữ liệu
data, labels = loadData('train.csv')
```

- Bước 2: Hiển thị hình ảnh theo nhãn
  Chuyển dữ liệu đọc được thành dữ liệu ảnh

```
# chuyển đổi data về dữ liệu hình ảnh grayscale
images = data2Img(data)
```

Hiển thị hình ảnh, cụ thể ở đây hiển thị 10 hình ảnh đầu tiên có label = 0

```
# hiển thị 10 tấm hình có label = 0
showImgByLabel(images, labels, 0, 10)
```



- Bước 3: Tiền xử lí dữ liệu

3.1 Các **kĩ thuật** dùng trong **xử lí dữ liệu**:

- Phân tích thành phần chính (PCA): là một thuật toán thống kê sử dụng phép biến đổi trực giao để biến đổi một tập hợp dữ liệu từ một không gian nhiều chiều sang một không gian mới ít chiều hơn nhằm tối ưu hóa việc thể hiện sự biến thiên của dữ liệu. Các ưu điểm của PCA:-
  - Giảm số chiều của không gian chứa dữ liệu khi nó có số chiều lớn.
  - Xây dựng những trục tọa độ mới, thay vì giữ lại các trục của không gian cũ, nhưng lại có khả năng biểu diễn dữ liệu tốt tương đương, và đảm bảo độ biến thiên của dữ liệu trên mỗi chiều mới.
  - Tạo điều kiện để các liên kết tiềm ẩn của dữ liệu có thể được khám phá trong không gian mới, mà nếu đặt trong không gian cũ thì khó phát hiện vì những liên kết này không thể hiện rõ.
  - Đảm bảo các trục tọa độ trong không gian mới luôn trực giao đôi một với nhau, mặc dù trong không gian ban đầu các trục có thể không trực giao.
- Chuẩn hóa dữ liệu (StandardScaler): nó sẽ biến đổi dữ liệu của bạn sao cho phân phối của nó sẽ có giá trị trung bình là 0 và độ lệch chuẩn là 1. Với phân phối dữ liệu, mỗi giá trị trung tập dữ liệu sẽ bị trừ cho giá trị trung bình mẫu, và sau đó chia cho độ lệch chuẩn của toàn bộ dữ liệu

$$z = \frac{x - \mu}{s}$$

3.2. Các bước thực hiện:

- Đào tạo các mô hình Scaler và PCA sau đó lưu lại

```
# đào tạo scaler
scaler = trainStandardScaler(data_copy)
# save model
saveModel(scaler, 'scaler.pkl')
# Apply transform
data_copy = scaler.transform(data_copy)
# đào tạo PCA dựa trên data đã scaler
pca = trainPCA(data_copy, 400)
```

```
# save model
saveModel(pca, 'pca.pkl')
```

Thực hiện tiền xử lí với cố mô hình Scaler và PCA đã đào tạo

```
# tiền xử lí dữ liệu train
X_train, Y_train = preprocessingData(train_img, train_lbl)
```

- Bước 4: Đào tạo mô hình và lưu mô hình

Đào tạo: dựa vào dữ liệu đã tiền xử lí dữ liệu ở Bước 3 để tiền hành đào tạo mô hình

```
# train mô hình hồi quy Logistic
model = trainLogisticRegression(X_train, Y_train)
```

Lưu mô hình

```
# save model
saveModel(model, 'logistic.pkl')
```

- Bước 5: Test mô hình dựa trên dữ liệu được tách ra từ bộ dữ liệu train.csv được cung cấp

```
# test model

# load model logistic
model_pre = loadModel('logistic.pkl')

# tiền xử lí dữ liệu test dựa trên scaler, PCA
X_test, Y_test = preprocessingData(test_img, test_lbl)

# tiến hành test với dữ liệu đã tiền xử lí
testLogisticRegression(X_test, Y_test, model_pre)
```

```
Accuracy of the model on the 5400 test images: 100 %
```

- Bước 6: Thực hiện train trên 100% dữ liệu từ tập train.csv và thực hiện test trên test.csv (sẽ được cung cấp sau)

```
# tiến hành test thực

path_test = 'test.csv'

# load data
X_test, Y_test = loadData(path_test)

# load model logistic
model_pre = loadModel('logistic.pkl')

# tiền xử lí dữ liệu test dựa trên scaler, PCA
X_test_pre, Y_test_pre = preprocessingData(X_test, Y_test)

# tiến hành test với dữ liệu đã tiền xử lí
testLogisticRegression(X_test, Y_test, model_pre)
```

4. Bảng đo độ chính xác của mô hình

Test trên bộ dữ liệu train.csv được chia 80% train, 20% test	Test trên cơ sở đã train trên tập train.csv và test trên tập test.csv
100%	