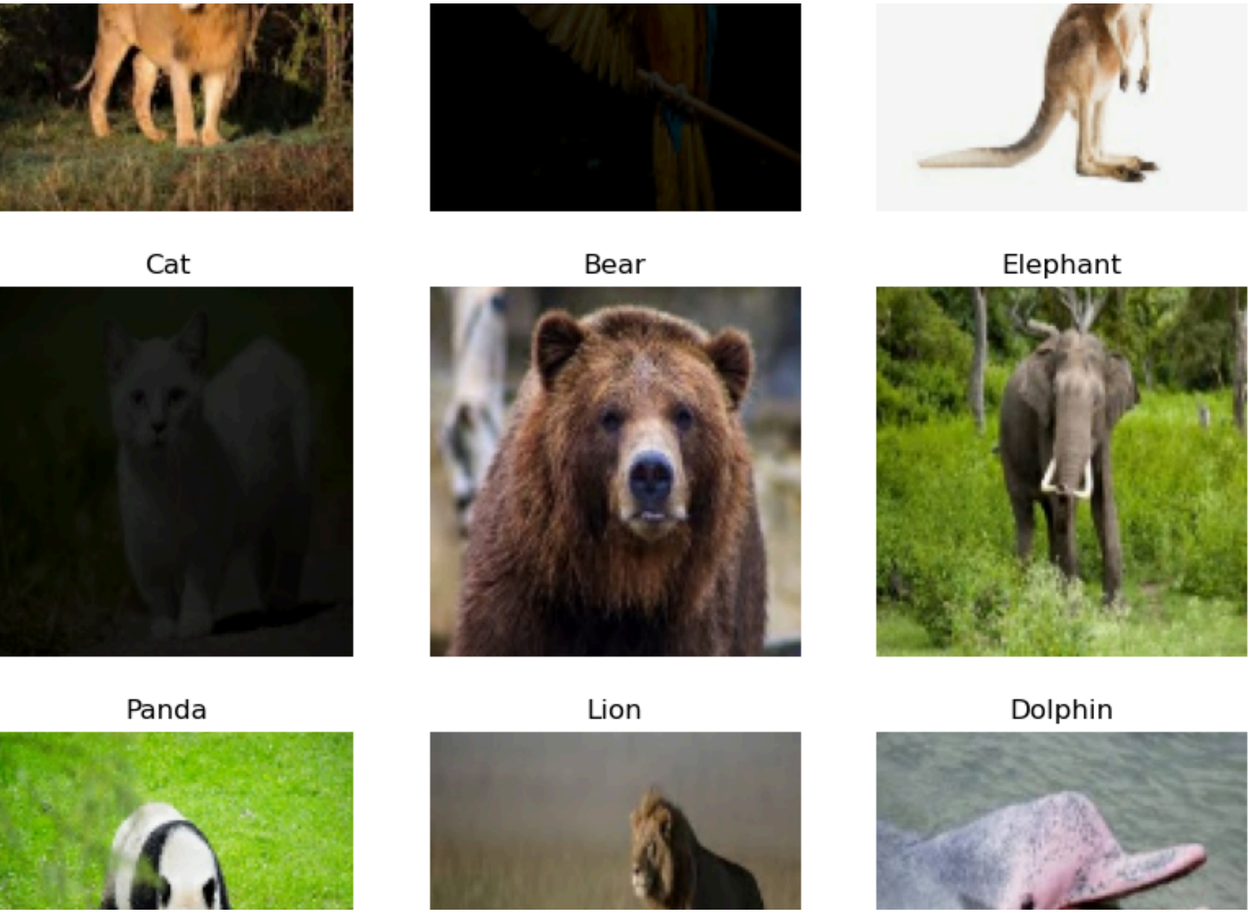
PROJECT REPORT

Nikhil Samuel Joseph J
. Internship ID: UMIP19644
Domain: Machine Learning

TOPICS CHOSEN

- Animal Classification
- Heart Disease Detection
- Mobile Phone Pricing Prediction
- Vehicle Prices Prediction



Animal Classification

- Creating a CNN model to classify animals based on the image.
- 'Bear', 'Bird', 'Cat', 'Cow', 'Deer', 'Dog', 'Dolphin', 'Elephant', 'Giraffe', 'Horse', 'Kangaroo', 'Lion', 'Panda', 'Tiger', and 'Zebra' are animals that are needed to be classified.

Model Architecture

- Created a random_brightness layer and rescaling layer to avoid overfitting and make it easier for the neural network to process the image.
- Two Convolution layer and a single pooling layer is used to reduce the number of pixels in the image.
- Finally, flatten the data and two dense layers to output the label of image processed.

Layer (type)	Output Shape	Param #
random_brightness (RandomBrightness)	(None, 128, 128, 3)	0
rescaling (Rescaling)	(None, 128, 128, 3)	0
conv2d (Conv2D)	(None, 126, 126, 64)	1,792
max_pooling2d (MaxPooling2D)	(None, 63, 63, 64)	0
conv2d_1 (Conv2D)	(None, 61, 61, 64)	36,928
flatten (Flatten)	(None, 238144)	0
dense (Dense)	(None, 128)	30,482,560
dense_1 (Dense)	(None, 15)	1,935

Total params: 30,523,215 (116.44 MB)

Trainable params: 30,523,215 (116.44 MB)

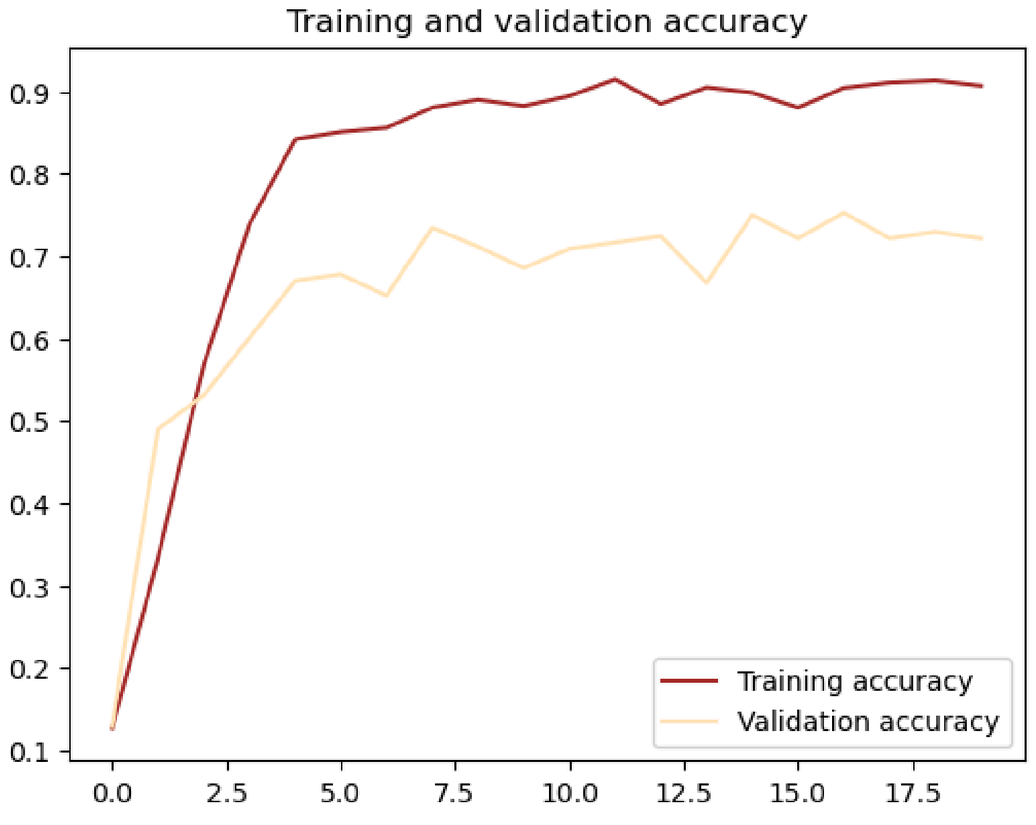
Non-trainable params: 0 (0.00 B)

```
Epoch 1/20
49/49 ━━━━━━━━━━━━━━━━━━━━ 19s 355ms/step - accuracy: 0.1041 - loss: 3.8886 - val_accuracy: 0.1289 - val_loss: 2.7173
Epoch 2/20
49/49 ━━━━━━━━━━━━━━━━━━━━ 14s 280ms/step - accuracy: 0.2711 - loss: 2.3377 - val_accuracy: 0.4897 - val_loss: 2.0316
Epoch 3/20
49/49 ━━━━━━━━━━━━━━━━━━━━ 14s 279ms/step - accuracy: 0.5589 - loss: 1.5580 - val_accuracy: 0.5309 - val_loss: 2.0427
Epoch 4/20
49/49 ━━━━━━━━━━━━━━━━━━━━ 14s 294ms/step - accuracy: 0.7266 - loss: 1.0606 - val_accuracy: 0.6005 - val_loss: 2.1985
Epoch 5/20
49/49 ━━━━━━━━━━━━━━━━━━━━ 14s 285ms/step - accuracy: 0.8303 - loss: 0.6278 - val_accuracy: 0.6701 - val_loss: 2.3877
Epoch 6/20
49/49 ━━━━━━━━━━━━━━━━━━━━ 14s 286ms/step - accuracy: 0.8588 - loss: 0.5360 - val_accuracy: 0.6778 - val_loss: 2.2454
Epoch 7/20
49/49 ━━━━━━━━━━━━━━━━━━━━ 14s 288ms/step - accuracy: 0.8537 - loss: 0.5442 - val_accuracy: 0.6521 - val_loss: 2.3587
Epoch 8/20
49/49 ━━━━━━━━━━━━━━━━━━━━ 14s 285ms/step - accuracy: 0.8720 - loss: 0.4949 - val_accuracy: 0.7345 - val_loss: 2.7960
Epoch 9/20
49/49 ━━━━━━━━━━━━━━━━━━━━ 14s 285ms/step - accuracy: 0.9022 - loss: 0.3538 - val_accuracy: 0.7113 - val_loss: 2.7820
Epoch 10/20
49/49 ━━━━━━━━━━━━━━━━━━━━ 14s 290ms/step - accuracy: 0.8875 - loss: 0.4499 - val_accuracy: 0.6856 - val_loss: 3.4893
Epoch 11/20
49/49 ━━━━━━━━━━━━━━━━━━━━ 14s 293ms/step - accuracy: 0.8899 - loss: 0.3777 - val_accuracy: 0.7088 - val_loss: 3.2257
Epoch 12/20
49/49 ━━━━━━━━━━━━━━━━━━━━ 14s 288ms/step - accuracy: 0.9287 - loss: 0.2921 - val_accuracy: 0.7165 - val_loss: 3.5409
Epoch 13/20
...
Epoch 19/20
49/49 ━━━━━━━━━━━━━━━━━━━━ 14s 295ms/step - accuracy: 0.9342 - loss: 0.2277 - val_accuracy: 0.7294 - val_loss: 3.6435
Epoch 20/20
49/49 ━━━━━━━━━━━━━━━━━━━━ 14s 283ms/step - accuracy: 0.9102 - loss: 0.3066 - val_accuracy: 0.7216 - val_loss: 4.4004
```

Training and evaluation

Trained the model for 20 epoch and created the model with training accuracy of 91%.

The graph shows the training and validation accuracy of the model.



Heart Disease Detection

- Creating a Logistic Regression model to classify if person has a heart disease or not based on the textual information.
- CSV file consist of 1190 records and 12 attributes.

Data Pre-processing

- First, remove the duplicate records
- Second, check for null values in dataframe.
- Finally, segregate labels (target) and other attributes seperately.

```
#Check for duplicates
x=df.duplicated()
print("Duplicated :",x.sum())

#drop duplicates

df=df.drop_duplicates()

df.duplicated().sum()
```

```
Duplicated : 272


0
```

```
#Check for null values

x=df.notnull().sum()
print(x)

#drop null values


df.dropna()
```

```
age                 918
sex                 918
chest pain type     918
resting bp s        918
cholesterol         918
fasting blood sugar 918
resting ecg         918
max heart rate      918
exercise angina     918
oldpeak             918
ST slope            918
target              918
dtype: int64
```

	age	sex	chest pain type	resting bp s	cholesterol	fasting blood sugar	resting ecg	max heart rate	exercise angina	oldpeak	ST slope	target
0	40	1	2	140	289	0	0	172	0	0.0	1	0
1	49	0	3	160	180	0	0	156	0	1.0	2	1
2	37	1	2	130	283	0	1	98	0	0.0	1	0
3	48	0	4	138	214	0	0	108	1	1.5	2	1
4	54	1	3	150	195	0	0	122	0	0.0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...
1185	45	1	1	110	264	0	0	132	0	1.2	2	1
1186	68	1	4	144	193	1	0	141	0	3.4	2	1
1187	57	1	4	130	131	0	0	115	1	1.2	2	1
1188	57	0	2	130	236	0	2	174	0	0.0	2	1
1189	38	1	3	138	175	0	0	173	0	0.0	1	0

Created and trained the model using sci-kit learn package and created the model with accuracy of 83% and good f1-score.

Training and evaluation

Usage of classification report function to show the performance of our model

```
#Model Training
model=LogisticRegression()
model.fit(X_train,y_train)
```

	precision	recall	f1-score	support
0.0	0.82	0.77	0.79	77
1.0	0.84	0.88	0.86	107
accuracy			0.83	184
macro avg	0.83	0.82	0.83	184
weighted avg	0.83	0.83	0.83	184

Mobile Phone Pricing Prediction

- Creating a K Nearest Neighbor model to classify the price of mobile phone based on price segments.
- CSV file consist of 2000 records and 21 attributes.
- Price segments are classified into 4 segments.

Data Pre-processing

- First, check for duplicate records and removing unwanted attributes from dataset.
- Attributes which are removed are 'm_dep', 'px_height', 'px_width', 'mobile_wt', 'sc_w', 'sc_h', 'talk_time' and, 'n_cores'
- Finally, check for null values. Segregate labels (target) and other attributes.

```
#Check for duplicates
x=df.duplicated()
print("Duplicated :",x.sum())

df=df.drop(columns=['m_dep','px_height','px_width','mobile_wt','sc_w','sc_h','talk_time','n_cores'])
df
```

```
Duplicated : 0
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	pc	ram	three_g	touch_screen	wifi	price_range
0	842	0	2.2	0	1	0	7	2	2549	0	0	1	1
1	1021	1	0.5	1	0	1	53	6	2631	1	1	0	2
2	563	1	0.5	1	2	1	41	6	2603	1	1	0	2
3	615	1	2.5	0	0	0	10	9	2769	1	0	0	2
4	1821	1	1.2	0	13	1	44	14	1411	1	1	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1995	794	1	0.5	1	0	1	2	14	668	1	1	0	0
1996	1965	1	2.6	1	0	0	39	3	2032	1	1	1	2
1997	1911	0	0.9	1	1	1	36	3	3057	1	1	0	3
1998	1512	0	0.9	0	4	1	46	5	869	1	1	1	0
1999	510	1	2.0	1	5	1	45	16	3919	1	1	1	3

2000 rows × 13 columns

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	pc	ram	three_g	touch_screen	wifi	price_range
0	842	0	2.2	0	1	0	7	2	2549	0	0	1	1
1	1021	1	0.5	1	0	1	53	6	2631	1	1	0	2
2	563	1	0.5	1	2	1	41	6	2603	1	1	0	2
3	615	1	2.5	0	0	0	10	9	2769	1	0	0	2
4	1821	1	1.2	0	13	1	44	14	1411	1	1	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1995	794	1	0.5	1	0	1	2	14	668	1	1	0	0
1996	1965	1	2.6	1	0	0	39	3	2032	1	1	1	2
1997	1911	0	0.9	1	1	1	36	3	3057	1	1	0	3
1998	1512	0	0.9	0	4	1	46	5	869	1	1	1	0
1999	510	1	2.0	1	5	1	45	16	3919	1	1	1	3

2000 rows × 13 columns

Created and trained the model using sci-kit learn package and created the model with accuracy of 81% and good f1-score.

Training and evaluation

Usage classification report function to show the performance of our model

```
#Model Training
model=KNeighborsClassifier(n_neighbors=4)
model.fit(X_train,y_train)
```

	precision	recall	f1-score	support
0.0	0.89	0.92	0.91	93
1.0	0.79	0.79	0.79	107
2.0	0.66	0.72	0.69	90
3.0	0.91	0.79	0.84	110
accuracy			0.81	400
macro avg	0.81	0.81	0.81	400
weighted avg	0.81	0.81	0.81	400

Vehicle Prices Prediction

- Creating a Random Forest Regression model to predict the price of mobile phone based on price segments.
- CSV file consist of 1002 records and 17 attributes.
- Price segments is predicted using this model.

Data Pre-processing

- First, check for duplicate records and removing unwanted attributes from dataset.
- Check for null values and remove them.
- Finally, we change categorical data to numerical data. Segregate labels (target) and other attributes.

```
lb_body=LabelEncoder()
lb_drivetrain=LabelEncoder()
lb_fuel=LabelEncoder()
lb_make=LabelEncoder()

  #Check for null values

  x=df.notnull().sum()
  print(x)

  #drop null values

  df=df.dropna()

  df=df.drop(df.columns[[9,10]],axis=1)
  df
```

```
make              978
price             955
cylinders         875
fuel              971
mileage           946
transmission      976
trim              977
body              975
doors             971
exterior_color    973
interior_color    940
drivetrain        978
dtype: int64
```

Training and evaluation

Created and trained the model using sci-kit learn package.

MSE and r2-score is used to show the performance of our model.

```
#Model Training
model=RandomForestRegressor()
model.fit(X_train,y_train)
```

```
Mean Squared Error: 119724218.85470778
R-squared: 0.6044892960284172
```

	make	price	cylinders	fuel	mileage	body	drivetrain
0	12	74600.0	6.0	4	10.0	6	1
1	12	50170.0	6.0	4	1.0	6	1
2	8	96410.0	8.0	4	0.0	6	1
3	6	46835.0	8.0	4	32.0	6	0
4	20	81663.0	6.0	0	10.0	5	1
...	...	...	...	...	...	...	...
996	20	69315.0	6.0	0	0.0	5	1
997	18	59037.0	4.0	0	10.0	0	3
999	12	69085.0	6.0	4	20.0	6	1
1000	19	43495.0	6.0	4	6.0	6	0
1001	4	48995.0	8.0	4	31.0	5	3

781 rows × 7 columns

Thank You