Use Case Test

Image Classification & Fraud Detection

Image Classification: Problem Statement

Build a classifier model to classify the images test set, given a dataset of 4 classes:

- Art & Culture
- Architecture
- Food and Drinks
- Travel and Adventure

Image Classification: Dataset

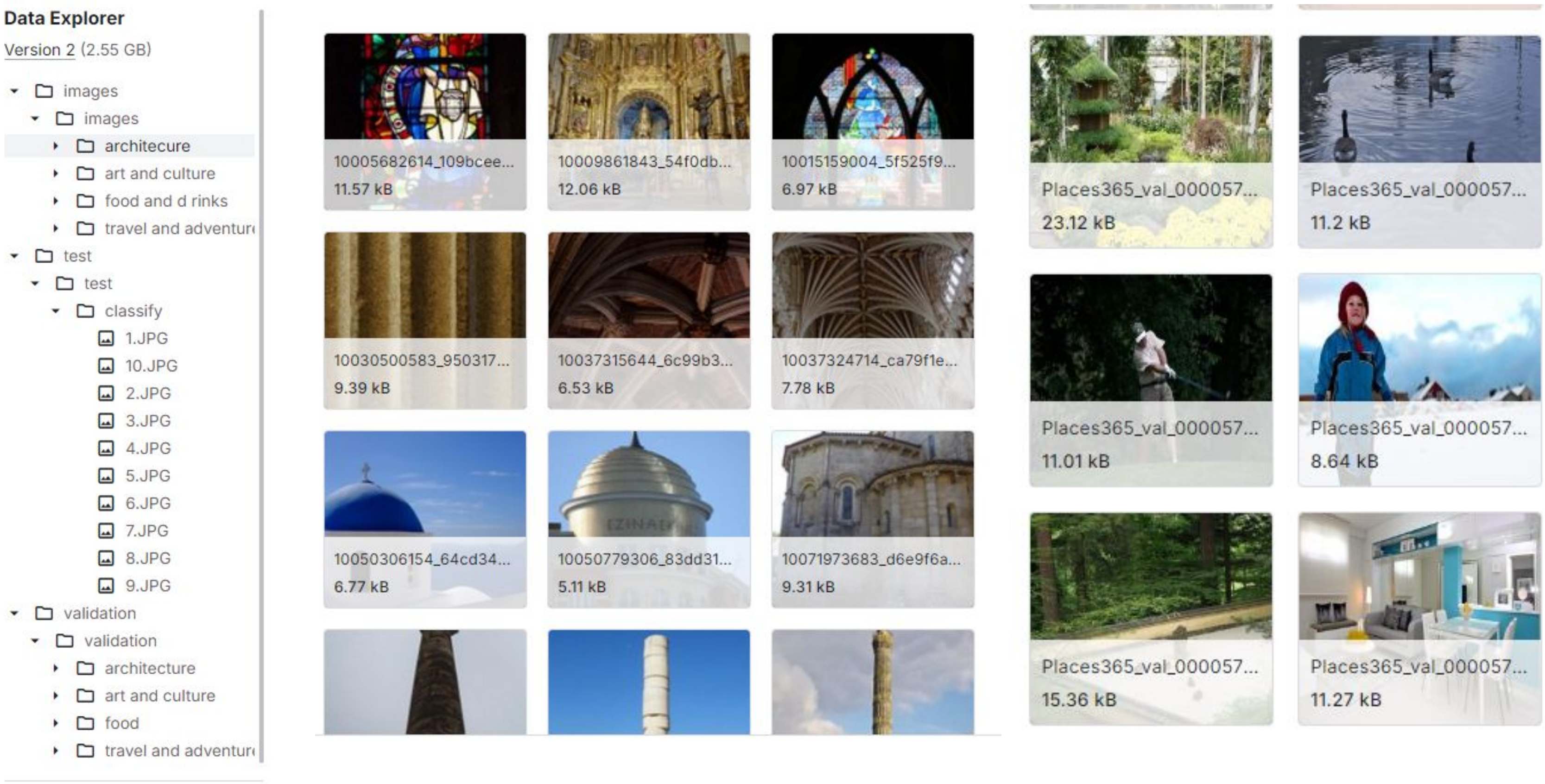


Image Classification: Load Batches from Dataset

```
train_dir = '/kaggle/input/image-classification/images/images'
validation_dir = '/kaggle/input/image-classification/validation/validation'
test_dir = '/kaggle/input/image-classification/test/test'
```

```
train_datagen = ImageDataGenerator(
    rescale=1./255,  # Normalise pixel
    rotation_range=20,  # Random rotation
    width_shift_range=0.2,  # Random horizontal shift
    height_shift_range=0.2,  # Random vertical shift
    shear_range=0.2,
    zoom_range=0.2,  # Random zoom
    horizontal_flip=True,  # Random flip
    fill_mode='nearest'
)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)
```

```
Found 35093 images belonging to 4 classes.
```

```
validation_datagen = ImageDataGenerator(rescale=1./255)

validation_generator = validation_datagen.flow_from_directory(
    validation_dir,
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)
```

```
Found 122 images belonging to 4 classes.
```

```
test_datagen = ImageDataGenerator(rescale=1./255)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=IMAGE_SIZE,
    batch_size=1,
    class_mode=None,
    shuffle=False
)
```

```
Found 10 images belonging to 1 classes.
```

Image Classification: Train Model

Version 1: Vanilla (manually define each layer)

```
vanilla_model = tf.keras.Sequential([

    tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', input_shape=(IMAGE_SIZE
[0], IMAGE_SIZE[1], 3)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),

    tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),

    tf.keras.layers.Flatten(),

    tf.keras.layers.Dense(1024, activation='relu'),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(4, activation='softmax')
])

vanilla_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=

vanilla_model.summary()
```

```
vanilla_model = tf.keras.Sequential([

    tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same', input_shape=(IMAGE_SIZE[0], IMAGE_SIZE[1], 3)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),

    tf.keras.layers.Conv2D(128, (3, 3), activation='relu', padding='same'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),

    tf.keras.layers.Flatten(),

    tf.keras.layers.Dense(1024, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(4, activation='softmax')
])

vanilla_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

vanilla_model.summary()
```

Image Classification: Evaluate Model

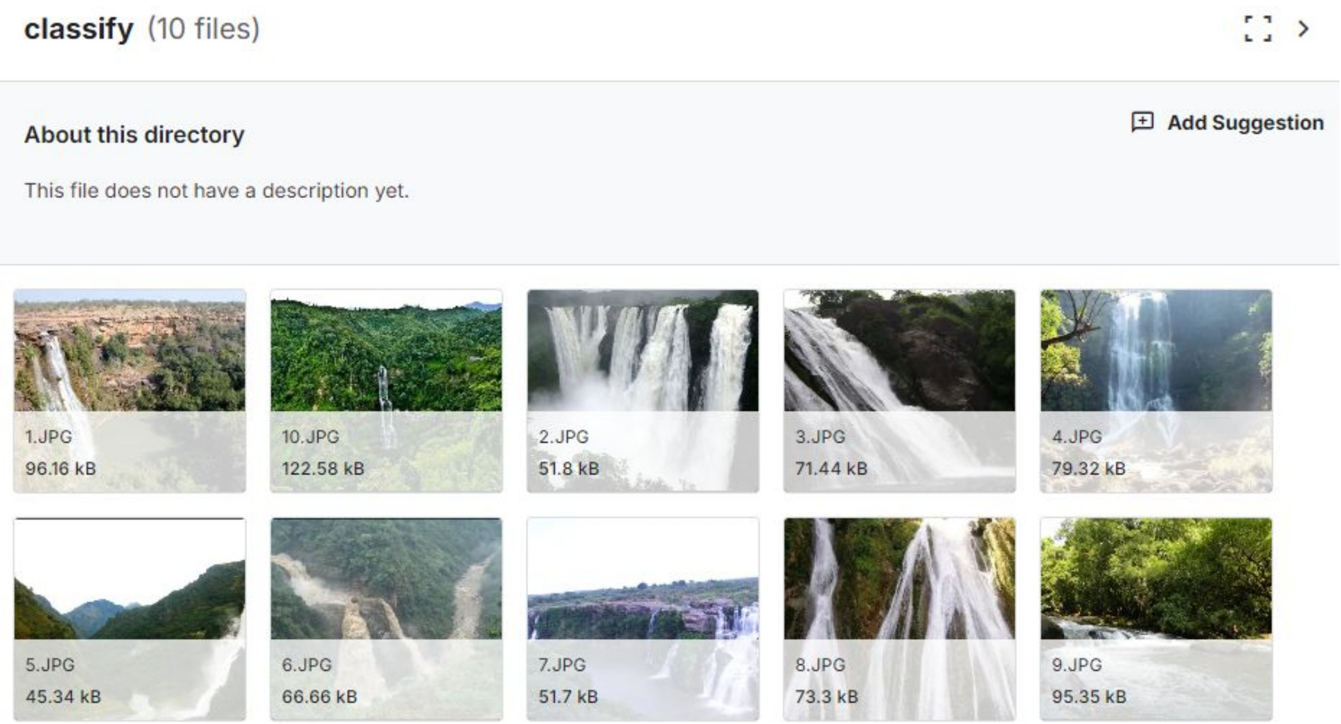


Image Classification: Evaluate Model

Version 1: Vanilla (manually define each layer)

At least in these 2 models, none are right

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step
          Filename Predicted Label
0   classify/1.JPG     architecure
1  classify/10.JPG     architecure
2   classify/2.JPG     architecure
3   classify/3.JPG     architecure
4   classify/4.JPG     architecure
5   classify/5.JPG     architecure
6   classify/6.JPG     architecure
7   classify/7.JPG     architecure
8   classify/8.JPG     architecure
9   classify/9.JPG     architecure
```

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step
          Filename  Predicted Label
0   classify/1.JPG  art and culture
1  classify/10.JPG  art and culture
2   classify/2.JPG  art and culture
3   classify/3.JPG  art and culture
4   classify/4.JPG  art and culture
5   classify/5.JPG  art and culture
6   classify/6.JPG  art and culture
7   classify/7.JPG  art and culture
8   classify/8.JPG  art and culture
9   classify/9.JPG  art and culture
```

Image Classification: Train Model

Version 2: Pretrained (VGG16)

```python
base_model = tf.keras.applications.VGG16(
    input_shape=(IMAGE_SIZE[0], IMAGE_SIZE[1], 3),
    include_top=False,
    weights='imagenet'
)

for layer in base_model.layers:
    layer.trainable = False

vgg16_model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(4, activation='softmax')  # 4 class
])

vgg16_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Check the architecture of the VGG16 model
vgg16_model.summary()
```

Layer (type)	Output Shape	Param #
vgg16 (Functional)	?	14,714,688
flatten_1 (Flatten)	?	0 (unbuilt)
dense_2 (Dense)	?	0 (unbuilt)

Image Classification: Evaluate Model

Version 2: Pretrained (VGG16)

Before hyperparameter tuning (left) and after (right)

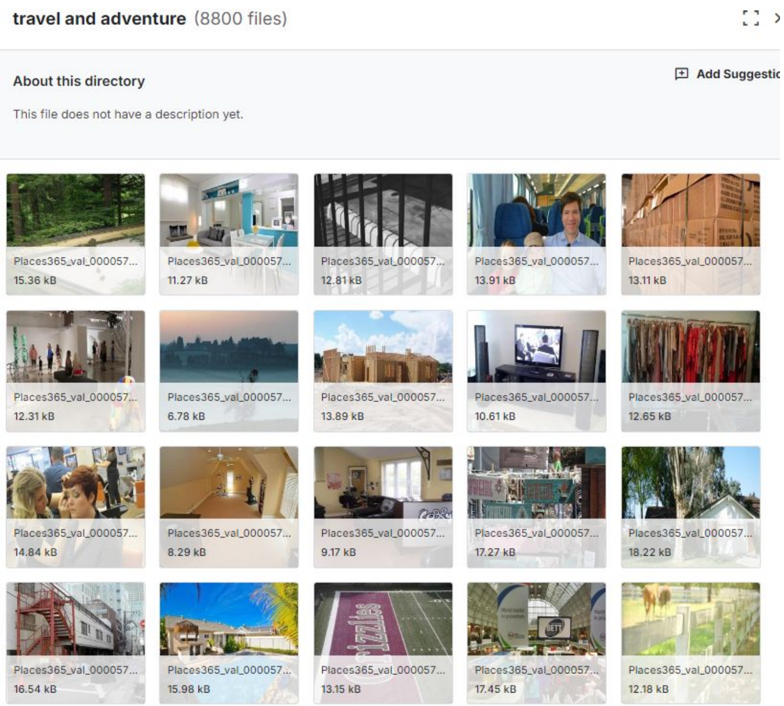
```
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step
          Filename        Predicted Label
0   classify/1.JPG        art and culture
1  classify/10.JPG        art and culture
2   classify/2.JPG        art and culture
3   classify/3.JPG        art and culture
4   classify/4.JPG        art and culture
5   classify/5.JPG  travel and  adventure
6   classify/6.JPG        art and culture
7   classify/7.JPG  travel and  adventure
8   classify/8.JPG        art and culture
9   classify/9.JPG  travel and  adventure
```

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step
          Filename        Predicted Label
0   classify/1.JPG  travel and  adventure
1  classify/10.JPG  travel and  adventure
2   classify/2.JPG        art and culture
3   classify/3.JPG        art and culture
4   classify/4.JPG  travel and  adventure
5   classify/5.JPG        art and culture
6   classify/6.JPG        art and culture
7   classify/7.JPG  travel and  adventure
8   classify/8.JPG  travel and  adventure
9   classify/9.JPG  travel and  adventure
```

Image Classification: Conclusion

The datasets are simply too varied. Although the test set should've been categorised into travel and adventure, the train set is extremely broad for identifying waterfalls are included in this class.

Suggestion: Use datasets that are categorised more specifically into subclasses



Fraud Detection: Problem Statement

Given a historical dataset of transactions, predict the likelihood of fraudulent online transactions using real-world e-commerce data to improve fraud detection accuracy and customer experience.

train_identity.csv (26.53 MB)

Detail Compact Column 41 of 41 columns

id_28	id_29	id_30	id_31	id_32
Found 53% New 45% Other (3255) 2%	Found 52% NotFound 46% Other (3255) 2%	[null] 46% Windows 10 15% Other (56410) 39%	chrome 63.0 15% mobile safari 11.0 9% Other (108810) 75%	0 – 32
New	NotFound	Android 7.0	samsung browser 6.2	32.0
New	NotFound	iOS 11.1.2	mobile safari 11.0	32.0
Found	Found		chrome 62.0	
New	NotFound		chrome 62.0	
Found	Found	Mac OS X 10_11_6	chrome 62.0	24.0
Found	Found	Windows 10	chrome 62.0	24.0
Found	Found	Android	chrome 62.0	32.0
Found	Found		chrome 62.0	
New	NotFound		chrome 62.0	

train_transaction.csv (683.35 MB)

Detail Compact Column 10 of 394 columns

TransactionID	isFraud	TransactionDT	TransactionAmt	ProductCD	card1
2.99m – 3.58m	0 – 1	86.4k – 15.8m	0.25 – 31.9k	W 74% C 12% Other (82351) 14%	1000
2987000	0	86400	68.5	W	13926
2987001	0	86401	29.0	W	2755
2987002	0	86469	59.0	W	4663
2987003	0	86499	50.0	W	18132
2987004	0	86506	50.0	H	4497
2987005	0	86510	49.0	W	5937
2987006	0	86522	159.0	W	12308
2987007	0	86529	422.5	W	12695
2987008	0	86535	15.0	H	2803
2987009	0	86536	117.0	W	17399
2987010	0	86549	75.887	C	16496
2987011	0	86555	16.495	C	4461

Fraud Detection: Read Dataset, Merge, Feature Engineering - day, hour, domain

```
train_transaction = pd.read_csv('/content/drive/MyDrive/Colab/Fraud Detection (1)/train_transaction.csv')
train_identity = pd.read_csv('/content/drive/MyDrive/Colab/Fraud Detection (1)/train_identity.csv')
test_transaction = pd.read_csv('/content/drive/MyDrive/Colab/Fraud Detection (1)/test_transaction.csv')
test_identity = pd.read_csv('/content/drive/MyDrive/Colab/Fraud Detection (1)/test_identity.csv')
```

```
[ ] train_df = pd.merge(train_transaction, train_identity, on='TransactionID', how='left')
    test_df = pd.merge(test_transaction, test_identity, on='TransactionID', how='left')
```

```
# day & hour
train_df['Transaction_day'] = train_df['TransactionDT'] // (24 * 60 * 60)
train_df['Transaction_hour'] = (train_df['TransactionDT'] // (60 * 60)) % 24
test_df['Transaction_day'] = test_df['TransactionDT'] // (24 * 60 * 60)
test_df['Transaction_hour'] = (test_df['TransactionDT'] // (60 * 60)) % 24

# domain
train_df['P_emaildomain'] = train_df['P_emaildomain'].str.split('.').str[-1]
train_df['R_emaildomain'] = train_df['R_emaildomain'].str.split('.').str[-1]
test_df['P_emaildomain'] = test_df['P_emaildomain'].str.split('.').str[-1]
test_df['R_emaildomain'] = test_df['R_emaildomain'].str.split('.').str[-1]
```

Fraud Detection: Feature Engineering - Fillna, Label encoding for categorical features

```
train_df.fillna(-999, inplace=True)
test_df.fillna(-999, inplace=True)
```

```
cat_cols = ['ProductCD', 'DeviceType', 'DeviceInfo', 'P_emaildomain', 'R_emaildomain'] + [f'card{i}' for i in range(1, 7)]

# Label encoding for categorical columns
for col in cat_cols:
    le = LabelEncoder()

    all_values = pd.concat([train_df[col], test_df[col]]).astype(str).unique()
    le.fit(all_values)

    train_df[col] = le.transform(train_df[col].astype(str))

del le
gc.collect()
```

Fraud Detection: Train/test split, model training

Why lgbm?

- Popular as being accurate on high dimensional dataset
- Fast, helps in re-training multiple times
- Handles unbalanced dataset

```python
params = {
    'objective': 'binary',
    'boosting_type': 'gbdt',
    'metric': 'auc',
    'is_unbalance': True,
    'learning_rate': 0.05,
    'num_leaves': 31,
    'device': 'cpu'
}

lgb_model = lgb.train(
    params,
    train_data,
    valid_sets=[train_data, val_data],
    num_boost_round=1000,
    categorical_feature=cat_cols
)
```

Fraud Detection: Result

Results in probability of being fraudulent, but the test files does not contain isFraud answers so unable to verify.

submission.csv available for consideration

```
TransactionID,isFraud
3663549,0.003141209229054317
3663550,0.0079958626246162
3663551,0.03064524630438539
3663552,0.004047690512886703
3663553,0.06279427527393197
3663554,0.08101031284726753
3663555,0.17079794387405645
3663556,0.2563131626518789
3663557,0.0016681028490185025
3663558,0.05108902685041743
3663559,0.1507509352727595
3663560,0.015731129791265598
3663561,0.40969529033532975
3663562,0.08623850882736579
3663563,0.03858052345531155
3663564,0.058640074612995335
3663565,0.25191143639483543
3663566,0.14888136324206883
3663567,0.49166257520077383
3663568,0.1268705019897579
```