Введение в нейронные сети

Урок 4. Сверточные нейронные сети

ДЗ - поиграться с кодом соревнования по Северстали, получить как можно лучший результат.
Альтернативное ДЗ - рассмотреть постановку и решение аналогичного индустриального кейса.
Сдавать как обычно блокнот.

Задания и ответы:

Эксперименты:

К уроку приложен архив с фотографиями, в котором было 351 изображение.

Посмотрим баланс данных (351)

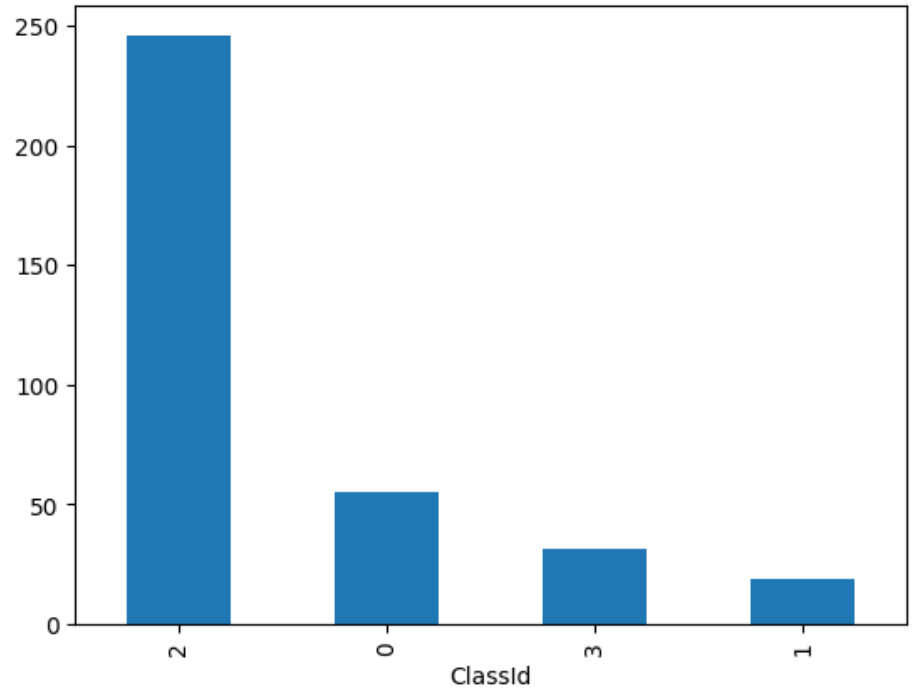
```
[292] df = pd.DataFrame(data=y, columns=["ClassId"])
      # df = df.convert_dtypes()
```

```
[293] df.isnull().sum()
```

```
ClassId    0
dtype: int64
```

```
df["ClassId"].value_counts().plot(kind = 'bar')
df["ClassId"].value_counts()
```

```
ClassId
2    246
0     55
3     31
1     19
Name: count, dtype: int64
```



В этом наборе больше всего меток второго класса

Борьба с дибалансом

Для борьбы с дисбалансом выполним случайное сэмплирование

т.е. выполним аугментацию случайным сэмплированием

```
[200] from imblearn.over_sampling import RandomOverSampler
      reshaped_X = X.reshape(X.shape[0],-1)
      oversample = RandomOverSampler(sampling_strategy='not majority')
      X, y  = oversample.fit_resample(reshaped_X , y)
      X = X.reshape(-1,120,120,3)
```

```
X.shape
```

```
(984, 120, 120, 3)
```

Проверим, действительно данные сбалансировались?

```
[202] y.shape
```

```
(984,)
```

```
y
```

```
array([1, 1, 1, 0, 1])
```

Посмотрим баланс получившихся данных (984)

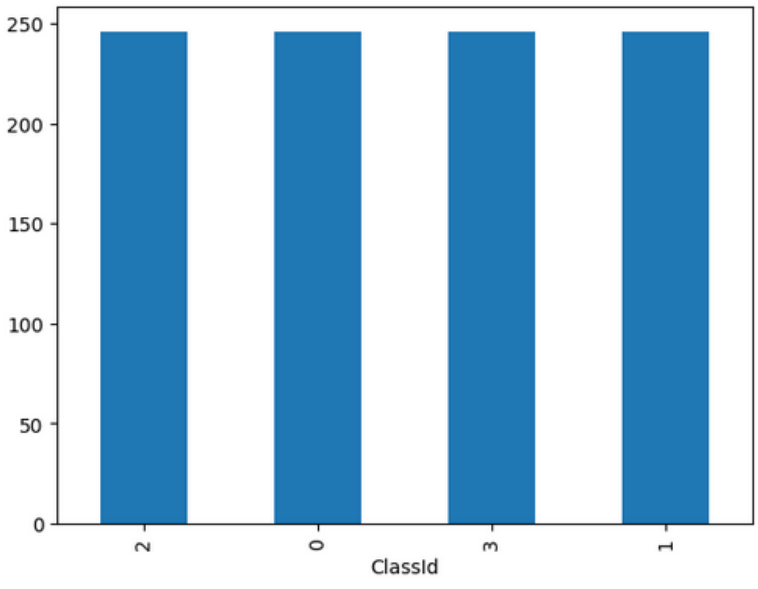
```
[204] df = pd.DataFrame(data=y, columns=["ClassId"])
      # df = df.convert_dtypes()
```

```
[205] df.isnull().sum()
```

```
ClassId    0
dtype: int64
```

```
df["ClassId"].value_counts().plot(kind = 'bar')
df["ClassId"].value_counts()
```

```
ClassId
2    246
0    246
3    246
1    246
Name: count, dtype: int64
```



Теперь данные сбалансированы

Тест 1

```
[327] model_1=Sequential()
      model_1.add(Conv2D(32,(3,3),input_shape=(120,120,3),activation="relu"))
      model_1.add(Flatten())
      model_1.add(Dense(4,activation="softmax"))
```

```
[328] model_1.compile(loss=keras.losses.categorical_crossentropy,
                      optimizer=keras.optimizers.Adam(),
                      metrics=["accuracy"])
```

```
history = model_1.fit(X_train,y_train,epochs=10,validation_data=(X_test,y_test),batch_size=128,
                      verbose=1)
```

```
Epoch 10/10
7/7 [==============================] - 0s 34ms/step - loss: 0.2745 - accuracy: 0.9390 - val_loss: 0.3978 - val_accuracy: 0.8629
```

Тест 2

```
[332] model_2=Sequential()
      model_2.add(Conv2D(32,(3,3),input_shape=(120,120,3),activation="relu"))
      model_2.add(MaxPooling2D(pool_size=(3,3)))
      model_2.add(Flatten())
      model_2.add(Dense(256,activation="relu"))
      model_2.add(Dense(4,activation="softmax"))
```

```
[333] model_2.compile(loss=keras.losses.categorical_crossentropy,
                      optimizer=keras.optimizers.Adam(),
                      metrics=["accuracy"])
```

```
history = model_2.fit(X_train,y_train,epochs=10,validation_data=(X_test,y_test),batch_size=128,
                      verbose=1)
```

```
Epoch 10/10
7/7 [==============================] - 0s 35ms/step - loss: 0.5275 - accuracy: 0.8145 - val_loss: 0.5951 - val_accuracy: 0.8274
```

Тест 3

```
[337] model_3=Sequential()
      model_3.add(Conv2D(32,(3,3),input_shape=(120,120,3),activation="relu"))
      model_3.add(MaxPooling2D(pool_size=(3,3)))
      model_3.add(Conv2D(64,(3,3),activation="relu"))
      model_3.add(MaxPooling2D(pool_size=(3,3)))
      model_3.add(Conv2D(64,(3,3),activation="relu"))
      model_3.add(MaxPooling2D(pool_size=(4,4)))
      model_3.add(Flatten())
      model_3.add(Dense(128,activation="relu"))
      model_3.add(Dropout(0.2))
      model_3.add(Dense(128,activation="relu"))
      model_3.add(Dropout(0.2))
      model_3.add(Dense(256,activation="relu"))
      model_3.add(Dense(4,activation="softmax"))
```

```
[338] early_stopping = tf.keras.callbacks.EarlyStopping(patience=5,min_delta=0.001,restore_best_weights=True)
```

Сборка и обучение модели

```
[339] model_3.compile(loss=keras.losses.categorical_crossentropy,
                      optimizer=keras.optimizers.Adam(),
                      metrics=["accuracy"])
```

```
[340] history = model_3.fit(X_train,y_train,epochs=10,validation_data=(X_test,y_test),batch_size=128,
                            verbose=1)
```

```
Epoch 10/10
7/7 [==============================] - 0s 44ms/step - loss: 0.9192 - accuracy: 0.5502 - val_loss: 0.9322 - val_accuracy: 0.5584
```

Итоговые результаты:

```
[343] result = model_3.evaluate(X_test, y_test)

      loss = result[0]
      accuracy = result[1]
      print(f"[+] Accuracy: {accuracy*100:.2f}%")
```

```
7/7 [==============================] - 0s 6ms/step - loss: 0.9322 - accuracy: 0.5584
[+] Accuracy: 55.84%
```

```
[344] result = model_2.evaluate(X_test, y_test)

      loss = result[0]
      accuracy = result[1]
      print(f"[+] Accuracy: {accuracy*100:.2f}%")
```

```
7/7 [==============================] - 0s 5ms/step - loss: 0.5951 - accuracy: 0.8274
[+] Accuracy: 82.74%
```

```
[345] result = model_1.evaluate(X_test, y_test)

      loss = result[0]
      accuracy = result[1]
      print(f"[+] Accuracy: {accuracy*100:.2f}%")
```

```
7/7 [==============================] - 0s 5ms/step - loss: 0.3978 - accuracy: 0.8629
[+] Accuracy: 86.29%
```