پویا شریفی 9823117

گزارشکار تمرین دوم عملیه ماشین لرنینگ

1)الف:

ما یک دیتا ست از یک سری اطلاعاتی که در یک hotel جمع اوری شده است را در دسترس داریم ,هدف ما این است که از روی این دیتا , Average daily rate را تخمین بزنیم.

یک دیتا ست برای train و validation به ما داده شده است و دیگری برای test .

میخواهیم که از یک mlp استفاده کنیم.

برای اینکه داده های categorical را به عدد تبدیل کنیم ,از لایبری پانداز استفاده میکنیم:

```
pd.Categorical
```

یک فانکشن به نام convert_to_cat زدم که در آن تمام دیتا های categorical را به عدد تبدیل میکنیم:

سپس از آن بخش train را میگیریم:

```
train_df,categorical_index=convert_to_cat(df)
```

شافل کردن دیتا ست:

```
train_df=train_df.sample(frac=1)
print(train_df)
```

جدا کردن دیتا ست:

میدانیم که میخواهیم ADR را پیشبینی کنیم,پس y ما همان ADR است,فقط چون دو وسط دیتا ست است ,مجبوریم که دیتا ست را دو تیکه کنیم,و سپس concat کنیم:

```
final_x_1=pd.DataFrame(train_df).iloc[:,:24]
final_x_2=(pd.DataFrame(train_df).iloc[:,25:])
final_x=pd.concat([final_x_1, final_x_2], axis="columns")
final_y=pd.DataFrame(train_df).iloc[:,24]
print(final_x)
```

نرمالایز کردن دیتا:

برای نرمالایز کردن دیتا سعی کردم که از لایبری استفاده نشود و فقط از std و میانگین دیتا استفاده کردم :

```
mean=final_x.mean()
standard_deviation=final_x.std()
final_x= (final_x - mean) / standard_deviation

print(final_x)
```

حالا به صورت sequential لایه هایمان را اضافه میکنیم:

```
model = keras.models.Sequential()
model.add(Dense(64, input_shape=input_shape, activation='relu'))

model.add(Dense(64, activation='relu'))

model.add(Dense(1))

# Configure the model and start training
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.01),
                    loss='mean_squared_error',
                    metrics=['MAE'])
history_1=model.fit((final_x), final_y, epochs=30, batch_size=64, verbose=1)
history_2=model.fit((final_x), final_y, epochs=50, batch_size=64, verbose=1)
# history_3=model.fit((final_x), final_y, epochs=100, batch_size=250, verbose=1)
```

حالا لایه ها را fit میکنیم,و میگذازیم که train شود:

1)ب:

حالا برای plot کردن loss این فانکشن را زدیم:

اول باید از history استفاده کنیم,تا اطلاعات را بیرون آورده و در یک لیست بزاریم:

```
loss_list = [s for s in history.history.keys() if 'loss' in s and 'val' not in s]
    val_loss_list = [s for s in history.history.keys() if 'loss' in s and 'val'
in s]
    acc_list = [s for s in history.history.keys() if 'acc' in s and 'val' not in
s]
    val_acc_list = [s for s in history.history.keys() if 'acc' in s and 'val' in
s]
    mae_list=[s for s in history.history.keys() if 'mean_absolute_error' in s and
'val' not in s]
    val_mae_list = [s for s in history.history.keys() if 'mean_absolute_error' in
s and 'val' in s]
```

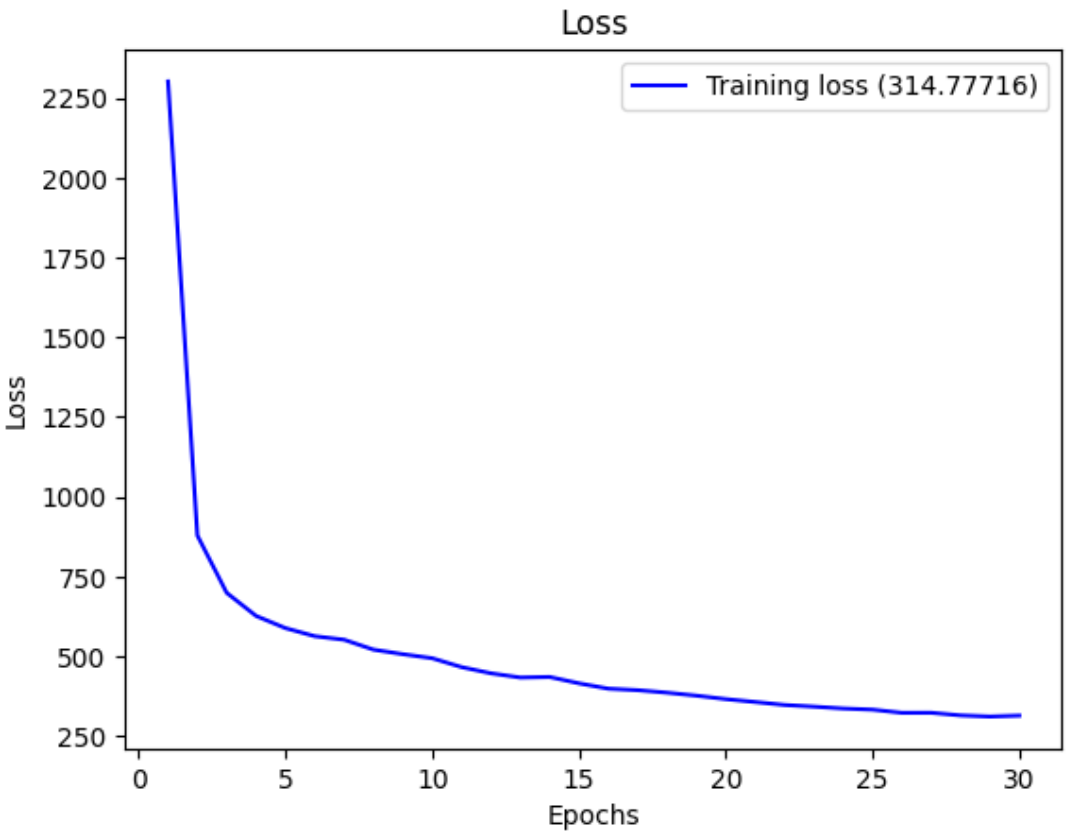
بعد هم باید پلات کنیم:

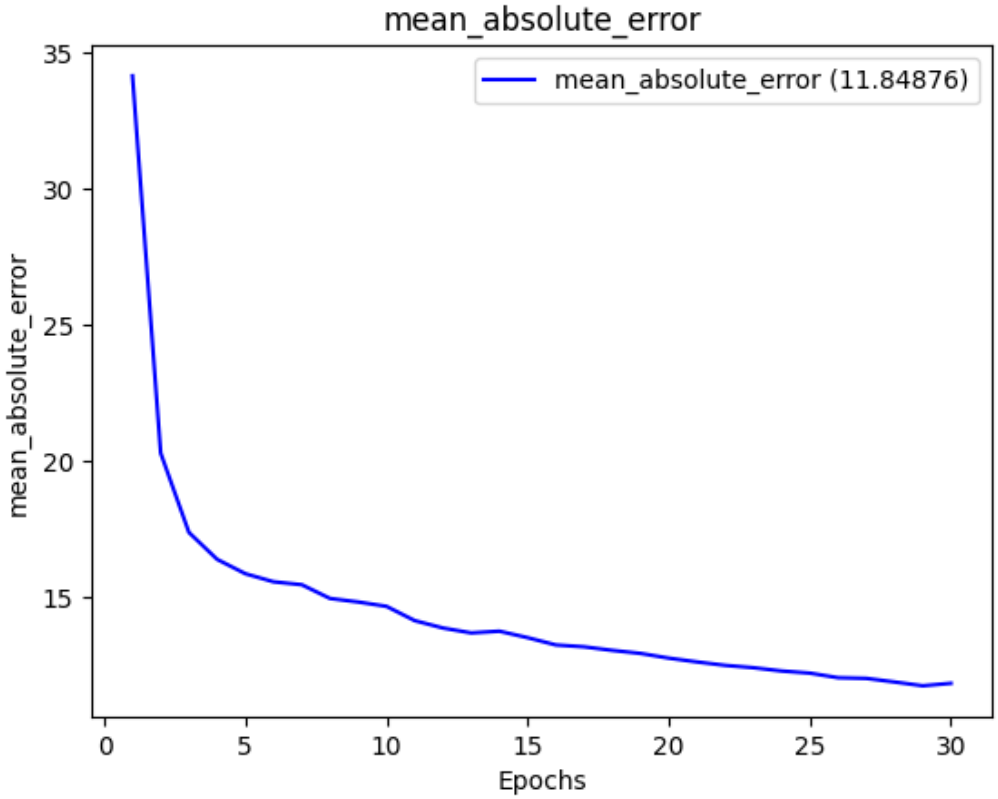
```
## Loss
    plt.figure(1)
    for l in loss_list:
        plt.plot(epochs, history.history[l], 'b', label='Training loss (' +
str(str(format(history.history[l][-1],'.5f'))+')'))
    for l in val_loss_list:
        plt.plot(epochs, history.history[l], 'g', label='Validation loss (' +
str(str(format(history.history[l][-1],'.5f'))+')'))


    plt.title('Loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()

    ## mae
    plt.figure(2)
    for l in mae_list:
        plt.plot(epochs, history.history[l], 'b', label='mean_absolute_error (' +
str(str(format(history.history[l][-1],'.5f'))+')'))
    # for l in val_acc_list:
    #     plt.plot(epochs, history.history[l], 'g', label='Validation accuracy ('
+ str(format(history.history[l][-1],'.5f'))+')')

    plt.title('mean_absolute_error')
    plt.xlabel('Epochs')
    plt.ylabel('mean_absolute_error')
    plt.legend()
    plt.show()
```

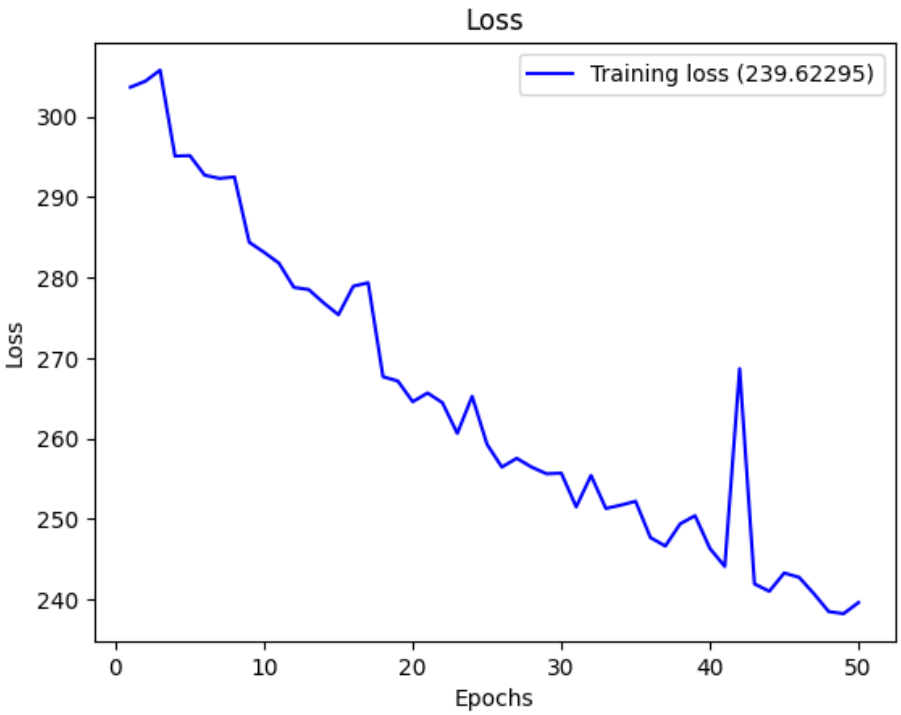
همه ی این کار ها را در یک فانکشن به اسم زیر قرار دادم:

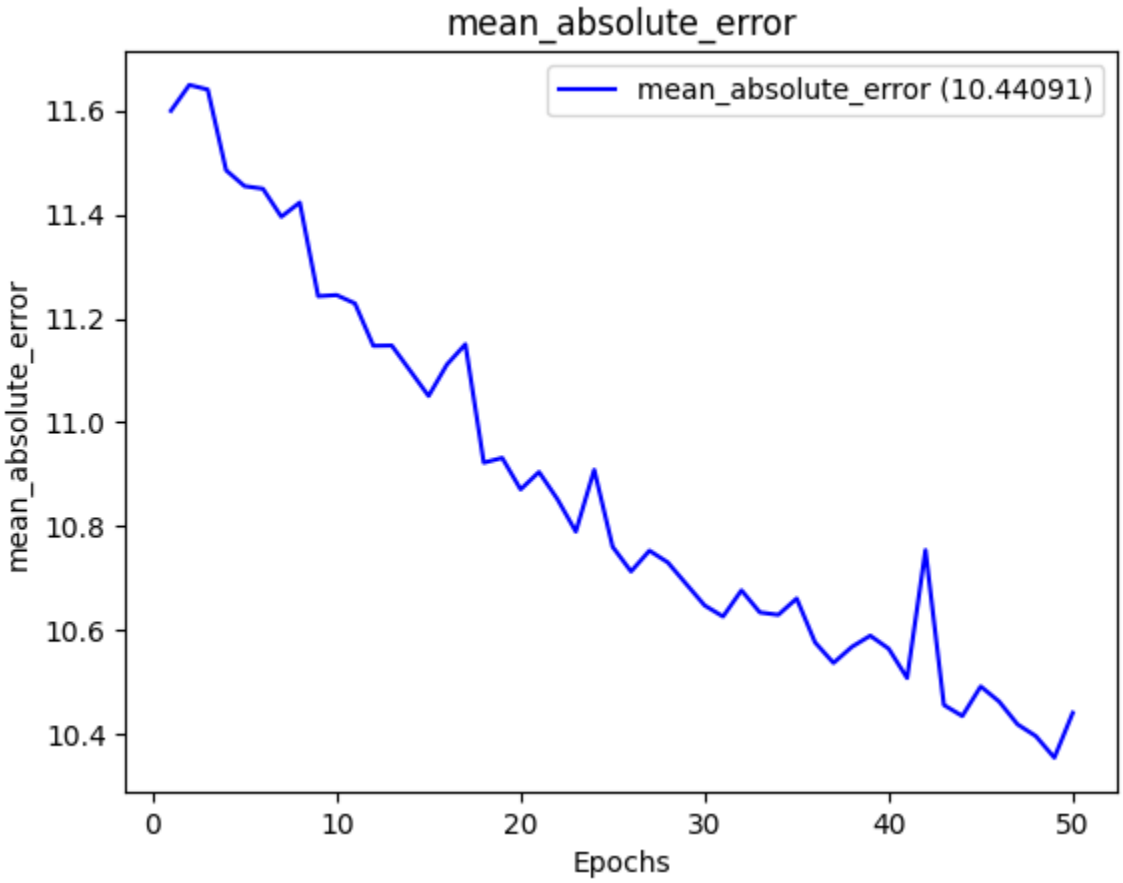
```
def plot_history(history):
```

figure های زیر را بدست آوردیم:


Loss
Training loss (314.77716)
Loss
Epochs
2250
2000
1750
1500
1250
1000
750
500
250
0
5
10
15
20
25
30


mean_absolute_error
mean_absolute_error (11.84876)
mean_absolute_error
Epochs
35
30
25
20
15
0
5
10
15
20
25
30


Loss
Training loss (239.62295)
300
290
280
270
260
250
240
Loss
0
10
20
30
40
50
Epochs


mean_absolute_error
mean_absolute_error (10.44091)
11.6
11.4
11.2
11.0
10.8
10.6
10.4
mean_absolute_error
0
10
20
30
40
50
Epochs

1)ج:

حالا باید دیتا ی تست رو وارد کنیم:

```
df_2 = pd.read_csv("H2.csv")
df_2=df_2.dropna()
test_df,categorical_index_2=convert_to_cat(df_2)
final_x_1_2=pd.DataFrame(test_df).iloc[:,:24]
final_x_2_2=(pd.DataFrame(test_df).iloc[:,25:])
test_x=pd.concat([final_x_1_2, final_x_2_2], axis="columns")
test_y=pd.DataFrame(test_df).iloc[:,24]
print(test_x)
```

```
mean=test_x.mean()
standard_deviation=test_x.std()
test_x= (test_x - mean) / standard_deviation
print(test_x)
```

حالا جواب ها را predict میکنیم و بعد آن ها را درون فایل save میکنیم:

```
y_pred=model.predict(test_x)
```

```
csv_write=pd.concat([pd.DataFrame(y_pred), pd.DataFrame(test_y)], axis="columns")
csv_write.to_csv("./comp.csv")
```

حالا مقدار اختلاف را به دو روش مختلف در آوردیم:

```
from sklearn.metrics import mean_squared_error
print(mean_squared_error(pd.DataFrame(y_pred), pd.DataFrame(test_y)))
model.evaluate(test_x,test_y)
```

نتیجه:

1)د:

برای این بخش از select k best, libarary استفاده کردیم,که خودش برایمان این کار را انجام میدهد:

```
from sklearn.feature_selection import SelectKBest
# from sklearn.feature_selection import chi2
from sklearn.feature_selection import SelectKBest, f_regression
select_reg =  SelectKBest(k=4, score_func=f_regression)
select_reg.fit(final_x, final_y)
X_train_housing_new = select_reg.transform(final_x)
X_train_housing_new.shape
```

```
kept_features = pd.DataFrame({'columns': final_x.columns,
                              'Kept': select_reg.get_support()})
kept_features
```

مثلا 10 تا مهم ترین feature, feature های زیر هستند:

	columns	Kept
0	IsCanceled	False
1	LeadTime	False
2	ArrivalDateYear	False
3	ArrivalDateMonth	True
4	ArrivalDateWeekNumber	True
5	ArrivalDateDayOfMonth	False
6	StaysInWeekendNights	False
7	StaysInWeekNights	False
8	Adults	True
9	Children	True
10	Babies	False
11	Meal	True
12	Country	False
13	MarketSegment	True
14	DistributionChannel	True
15	IsRepeatedGuest	False
16	PreviousCancellations	False
17	PreviousBookingsNotCanceled	False
18	ReservedRoomType	True
19	AssignedRoomType	True
20	BookingChanges	False
21	DepositType	False
22	DaysInWaitingList	False
23	CustomerType	False
24	RequiredCarParkingSpaces	False

(2

ما چون کلاس مقدمه ی هوش داریم ,این بخش رو یکبار انجام داده بودیم ,پس کمی خلاصه توضیح میدم و بیشتر فقط نتایج رو میزارم.

این لایه های ماست:

```
 # return model
    model = tf.keras.Sequential()

    # VGG_1
    model.add(tf.keras.layers.Conv2D(32, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same', input_shape=input_shape))
    model.add(tf.keras.layers.BatchNormalization())
    model.add(tf.keras.layers.Conv2D(32, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same', input_shape=input_shape))
    model.add(tf.keras.layers.BatchNormalization())
    model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
    model.add(tf.keras.layers.Dropout(0.2))

    # Output
    model.add(tf.keras.layers.Flatten())
    model.add(tf.keras.layers.Dense(128, activation='relu',
kernel_initializer='he_uniform'))
    model.add(tf.keras.layers.BatchNormalization())
    model.add(tf.keras.layers.Dropout(0.5))
    model.add(tf.keras.layers.Dense(10, activation='softmax'))
    return model
```

یا همان»

Model: "sequential_5"

_____

Layer (type) Output Shape Param #

==========

conv2d_8 (Conv2D) (None, 32, 32, 32) 896

_____

batch_normalization_9 (Batch (None, 32, 32, 32) 128

_____

conv2d_9 (Conv2D) (None, 32, 32, 32) 9248

_____

batch_normalization_10 (Batc (None, 32, 32, 32) 128

_____

max_pooling2d_5 (MaxPooling2 (None, 16, 16, 32) 0

_____

dropout_6 (Dropout) (None, 16, 16, 32) 0

_____

flatten_5 (Flatten) (None, 8192) 0

_____

dense_12 (Dense) (None, 128) 1048704

_____

batch_normalization_11 (Batc (None, 128) 512

_____

dropout_7 (Dropout) (None, 128) 0

_____

dense_13 (Dense) (None, 10) 1290

==========

Total params: 1,060,906

Trainable params: 1,060,522

Non-trainable params: 384

حالا fit میکنیم لایه ها را با مدلمون:

Train on 50000 samples

Epoch 1/6

50000/50000 [==========] - 134s 3ms/sample - loss: 0.7869 - acc: 0.7257

Epoch 2/6

50000/50000 [==============================] - 134s 3ms/sample - loss: 0.7215 - acc: 0.7485

Epoch 3/6

50000/50000 [==============================] - 133s 3ms/sample - loss: 0.6714 - acc: 0.7658

Epoch 4/6

50000/50000 [==============================] - 134s 3ms/sample - loss: 0.6317 - acc: 0.7785

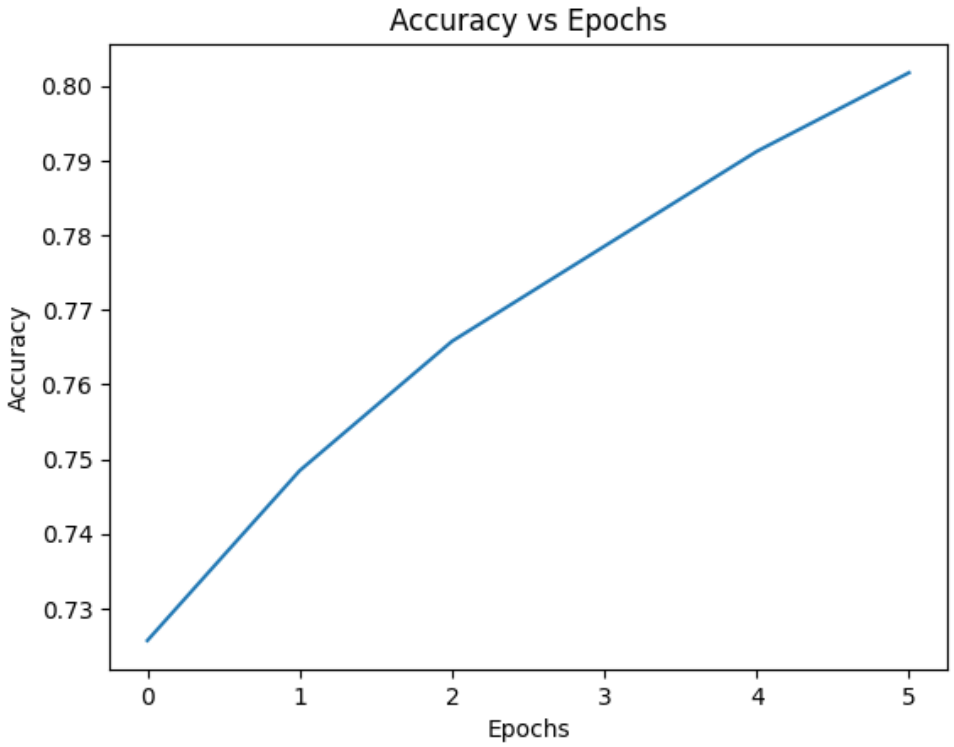
Epoch 5/6

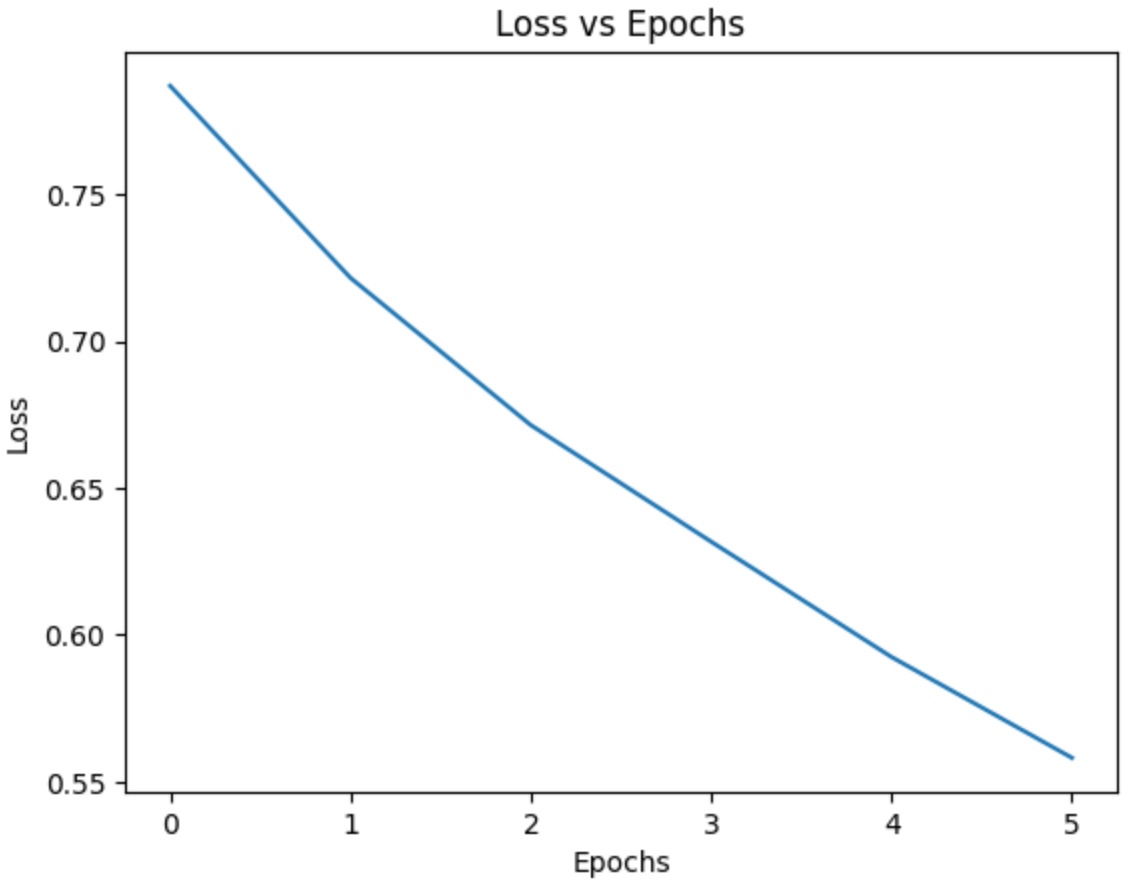
50000/50000 [==============================] - 128s 3ms/sample - loss: 0.5924 - acc: 0.7912

Epoch 6/6

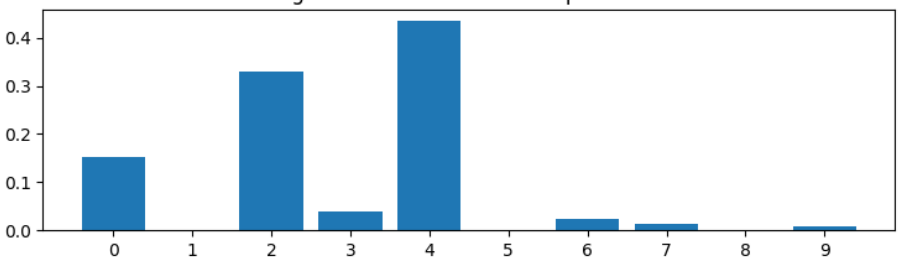
50000/50000 [==============================] - 140s 3ms/sample - loss: 0.5581 - acc: 0.8017

حالا نمودار ها را میکشیم:

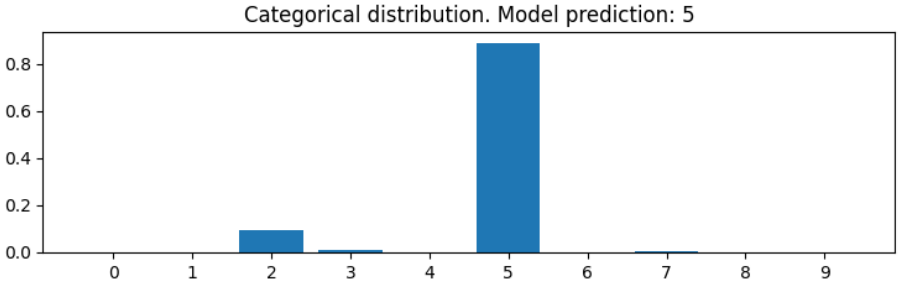



Loss vs Epochs
0.75
0.70
0.65
0.60
0.55
Loss
0
1
2
3
4
5
Epochs

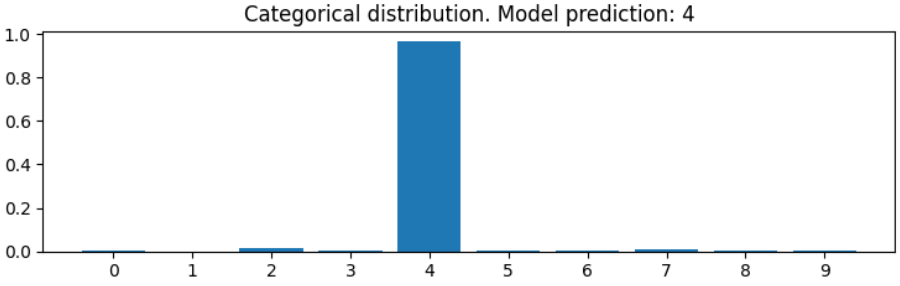
Digit [2]



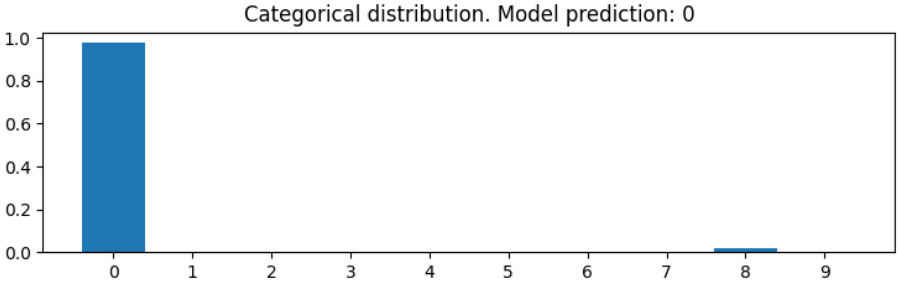
Digit [5]



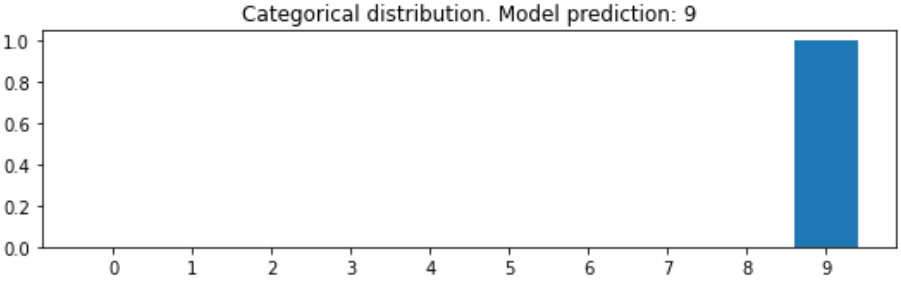
Digit [4]



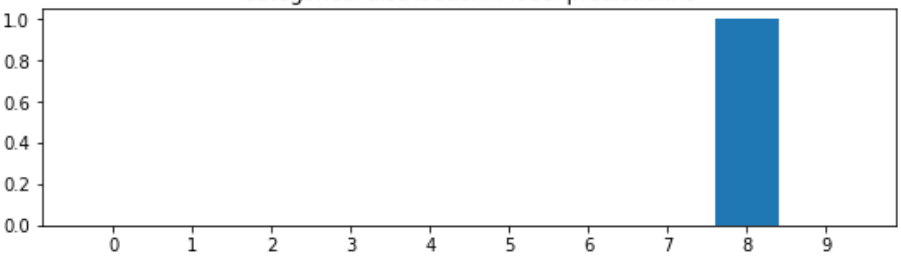
Digit [0]



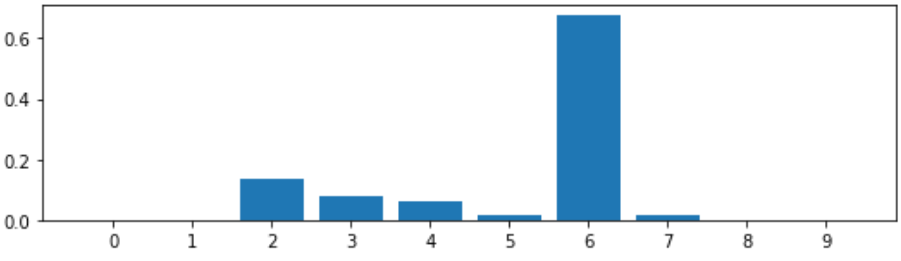
Digit [9]



Digit [8]



Digit [2]



Digit [5]