HORSES OR HUMANS: BINARY CLASSIFICATION

Dataset: a collection of 300 x 300 images, created by Laurence Moroney, Lead AI Advocate at Google.

Objective: train a Support Vector Machine to classify whether an image is of a horse or a human.

Image classification

THE DATASET: RENDERED IMAGES




'Emphasis has been taken to ensure diversity of humans, and to that end there are both men and women as well as Asian, Black, South Asian and Caucasians present in the training set'

- Photoreal CGI of various species of horse and in a variety of poses and locations
- Photoreal CGI of humans in a variety of poses and locations.

Train set

500 images of horses

527 images of humans

Test set

128 images of horses

128 images of humans

Binary Classification

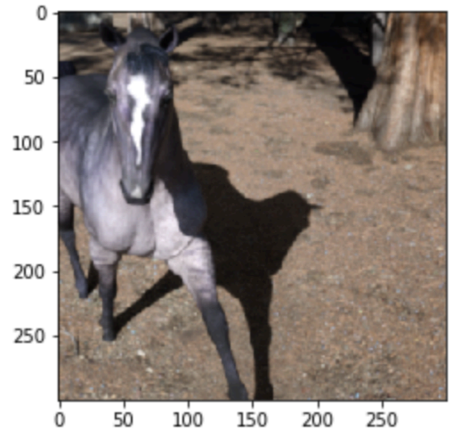
Human = 1

Horse = 0

EDA & DATA PREPROCESSING

```python
# load image as pixel array
data = image.imread('./horse-or-human/horses/horse43-5.png')

# summarize shape of the pixel array
print(data.dtype)
print(data.shape)

# display the array of pixels as an image
plt.imshow(data)
plt.show()
```

```
float32
(300, 300, 4)
```




```python
# convert image into a numpy array
img = np.ravel(data)

img
```

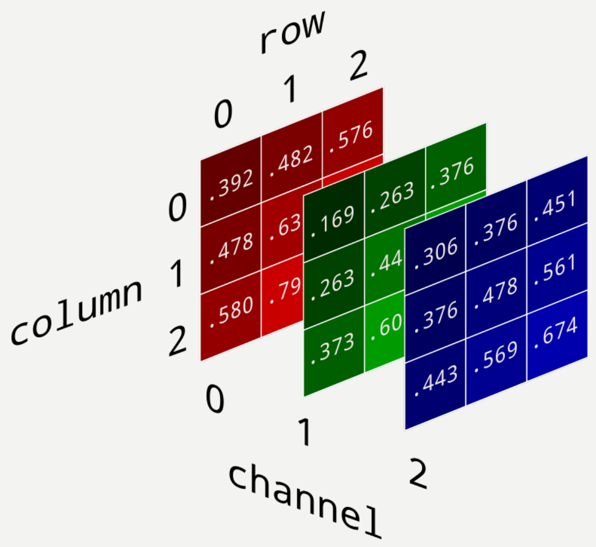
```
array([0.44705883, 0.4392157 , 0.41568628, ..., 0.5019608 , 0.42745098,
       1.        ], dtype=float32)
```

```python
# insert numpy array into a dataframe
df = pd.DataFrame([img])

df
```

3	4	5	6	7	8	9	...	359990	359991	359992	359993	359994	359995	359996	359997	359998	359999
1.0	0.435294	0.431373	0.419608	1.0	0.415686	0.415686	...	0.447059	1.0	0.584314	0.505882	0.431373	1.0	0.580392	0.501961	0.427451	1.0

Each image in the dataset is 300 x 300 x 4 (height x width x channels)

Each image is a tensor made up of 4 channels: red, green, blue and alpha

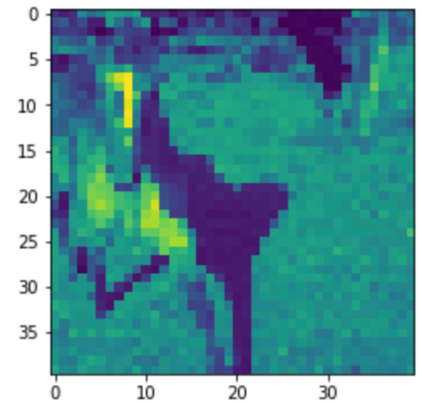


FEATURE EXTRACTION & DATAFRAMES

Using the cv2 Python library:

- Convert the images to grayscale to extract the most import features
- Resize images from 300 x 300 to 40 x 40
- Flatten the images to create 1-D array representations for the train set dataframe and test set dataframe
- Each image now a 1-D array of 1 x 1600

```
# Using cv2.imread() method
# Using 0 to read image in grayscale mode
img = cv2.imread('./horse-or-human/horses/horse43-5.png', 0)

IMG_SIZE = 40

resized_img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))

resized_img

plt.imshow(resized_img)
plt.show()
```



First five rows of the train set dataframe

	0	1	2	3	4	5	6	7	8	9	...	1592	1593	1594	1595	1596	1597	1598	1599	category	filename
0	43	27	39	124	90	112	151	145	137	148	...	105	73	117	113	114	110	114	136	0	horse43-5.png
1	156	157	157	157	157	158	158	157	157	157	...	183	182	213	201	183	196	206	214	0	horse06-5.png
2	141	141	140	140	140	139	139	139	139	138	...	42	32	29	115	178	177	174	173	0	horse20-6.png
3	234	194	180	177	180	174	173	170	169	166	...	177	179	176	178	184	192	200	196	0	horse04-7.png
4	109	110	110	111	109	109	110	115	124	136	...	46	54	49	45	48	42	60	32	0	horse41-7.png

TRAINING & TESTING

```
clr.fit(X_train, y_train)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=True, random_state=None, shrinking=True, tol=0.001,
    verbose=False)
```

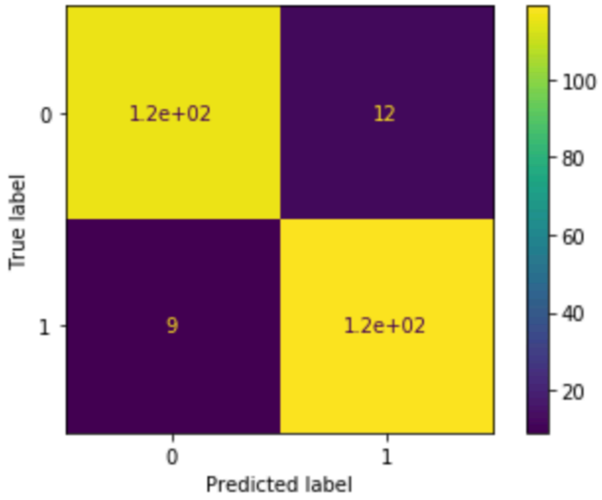
TRAIN SET DATAFRAME

```
features = train_df.columns[:-2].tolist()
target = "category”

X_train = train_df[train_features]
y_train = train_df[target]
```

TEST SET DATAFRAME

```
features = test_df.columns[:-2].tolist()
target = "category”

X_test = test_df[test_features]
y_test = test_df[target]
```

EVALUATION




```
from sklearn import metrics

predicted = clr.predict(X_test)

actual = y_test

metrics.accuracy_score(predicted, actual)
```

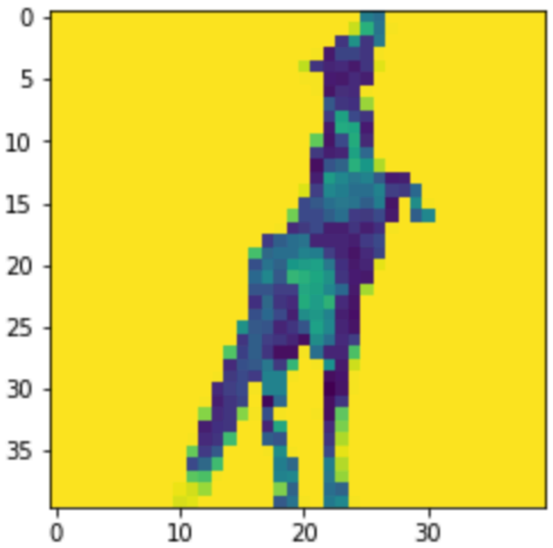
0.91796875

```
print(metrics.classification_report(actual, predicted))
```

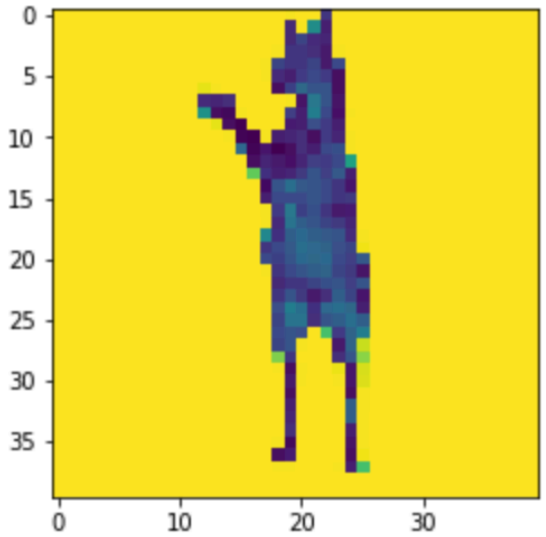
```
              precision    recall  f1-score   support

           0       0.93      0.91      0.92       128
           1       0.91      0.93      0.92       128

    accuracy                           0.92       256
   macro avg       0.92      0.92      0.92       256
weighted avg       0.92      0.92      0.92       256
```

	category	predicted	filename
2	0	1	horse3-498.png
31	0	1	horse2-412.png
33	0	1	horse5-123.png
35	0	1	horse2-201.png
46	0	1	horse2-582.png
48	0	1	horse2-596.png
74	0	1	horse4-503.png
77	0	1	horse4-501.png
89	0	1	horse3-484.png
101	0	1	horse1-554.png
107	0	1	horse3-521.png
117	0	1	horse4-548.png
137	1	0	valhuman03-00.png
138	1	0	valhuman04-09.png
194	1	0	valhuman02-17.png
199	1	0	valhuman02-16.png
227	1	0	valhuman01-24.png
235	1	0	valhuman03-23.png
241	1	0	valhuman01-23.png
245	1	0	valhuman04-10.png
255	1	0	valhuman03-24.png

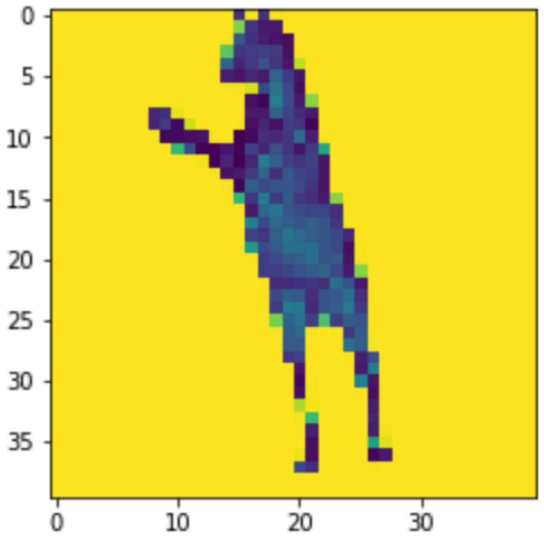
PREDICTED AS HUMAN: horse5-123.png



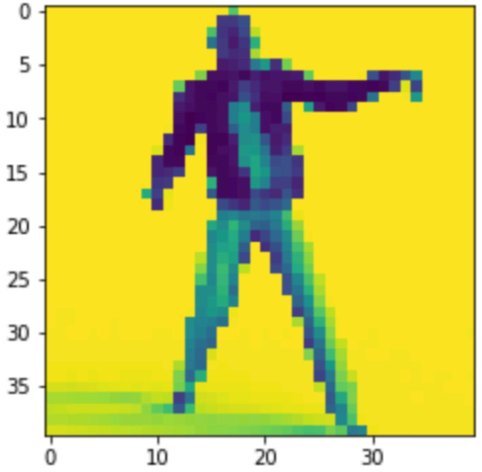
PREDICTED AS HUMAN: horse3-484.png



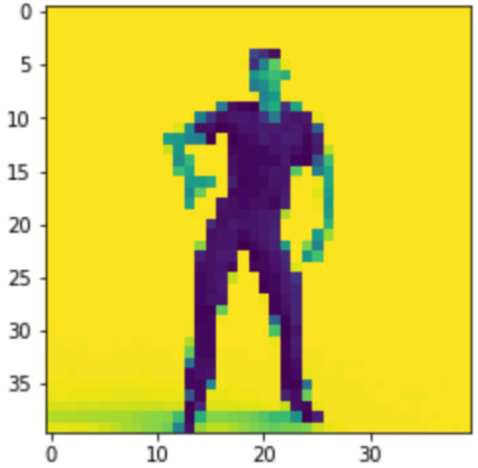
PREDICTED AS HUMAN: horse3-521.png



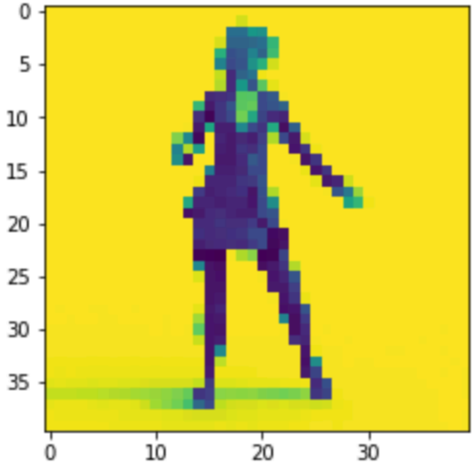
PREDICTED AS HORSE: valhuman03-23.png

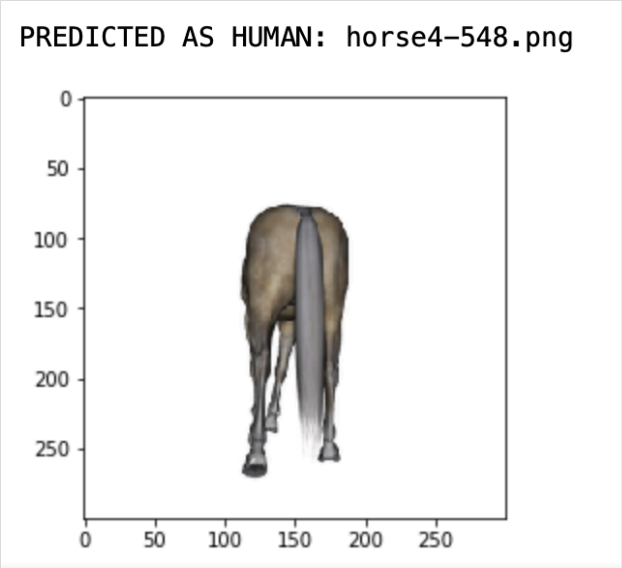


PREDICTED AS HORSE: valhuman01-23.png



PREDICTED AS HORSE: valhuman02-17.png

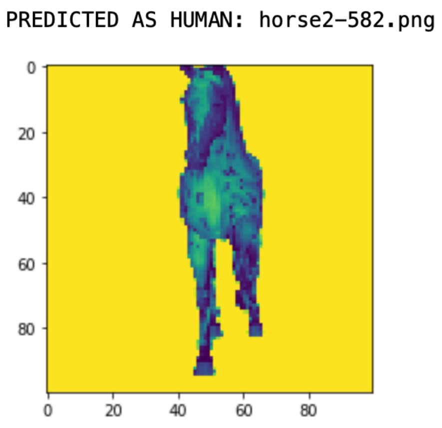


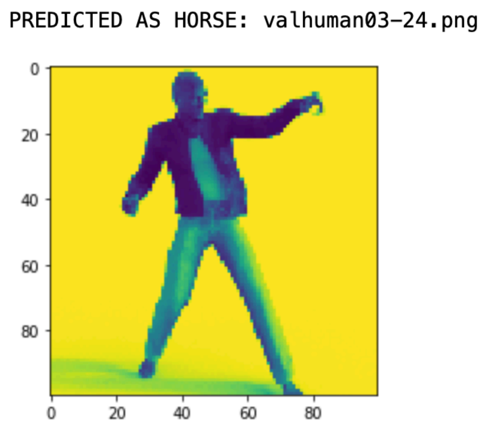


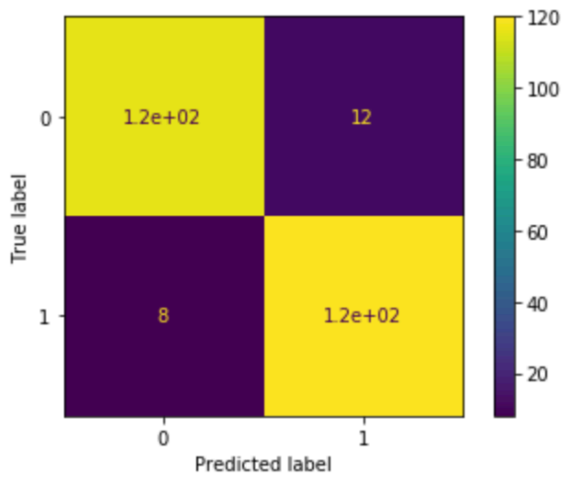
FURTHER EXPLORATION

IMG_SIZE = 100

ACC_SCORE =0.921875

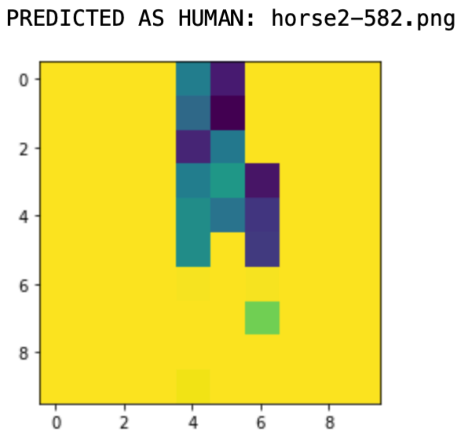


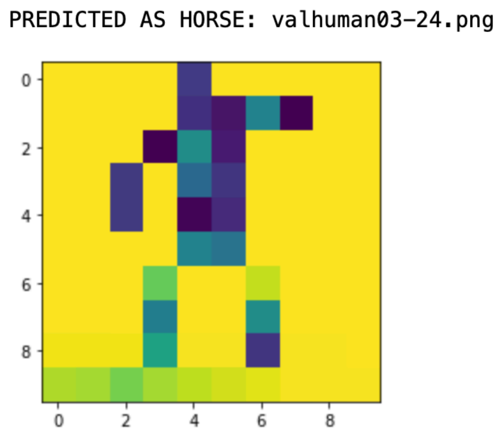


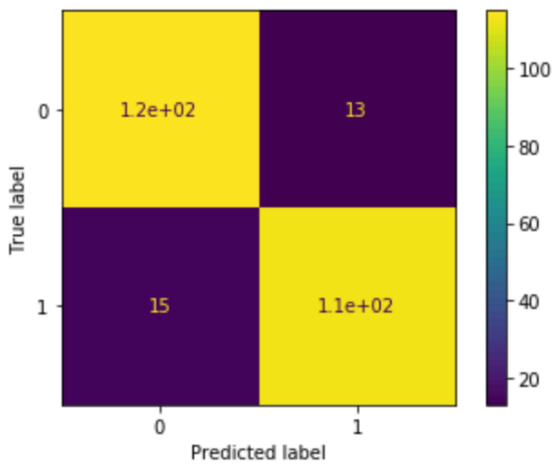


IMG_SIZE = 10

ACC_SCORE =0.890625



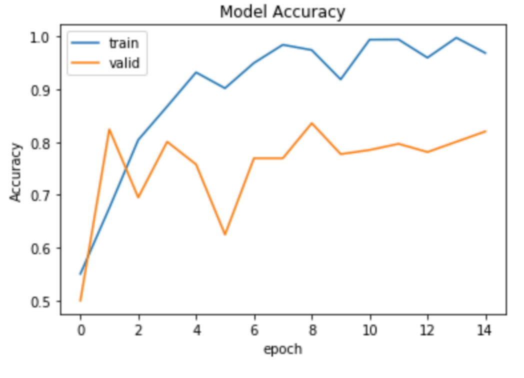


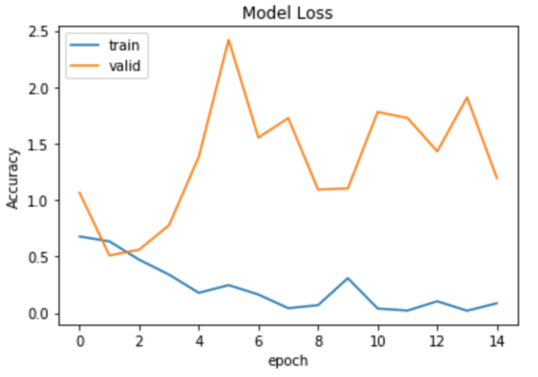


POST GENERAL ASSEMBLY

CNN

```
cnn_model = keras.models.Sequential([
    keras.layers.Conv2D(16, (3,3), activation="relu", input_shape=(300,300,3)),
    keras.layers.MaxPooling2D(2,2),

    keras.layers.Conv2D(16, (3,3), activation="relu"),
    keras.layers.MaxPooling2D(2,2),

    keras.layers.Conv2D(16, (3,3), activation="relu"),
    keras.layers.MaxPooling2D(2,2),

    keras.layers.Conv2D(16, (3,3), activation="relu"),
    keras.layers.MaxPooling2D(2,2),

    keras.layers.Conv2D(16, (3,3), activation="relu"),
    keras.layers.MaxPooling2D(2,2),

    # flatten the results
    keras.layers.Flatten(),
    # 512 neuron hidden layer
    keras.layers.Dense(512,activation="relu"),
    # outputs either 0 for "horses" or 1 for "humans"
    keras.layers.Dense(1,activation="sigmoid")
])
```





CNNs

- Practise and improve training neural nets

GANs

- Learn about Generative Adversarial Networks and the production of more data