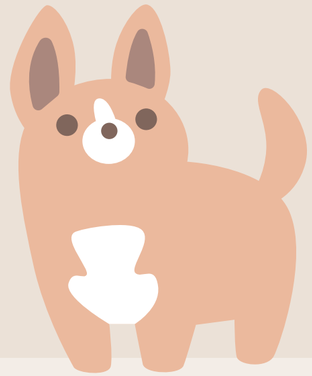
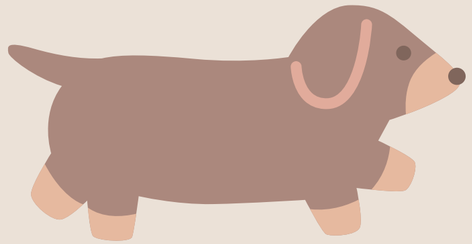
PAWSITIVE EMOTIONS

JUAN CAMILO MANTILLA - 2202050
ALFREDO GUTIERREZ NIETO - 2200137
MAXIMILIANO CORREA PICO - 2161594

TABLA DE CONTENIDO



PLANTEAMIENTO DEL PROYECTO



Este proyecto nace con la intención de poder hacer uso de las herramientas vistas en clase y de esta manera crear una IA que nos permita identificar las emociones que puede estar reflejando una mascota (perro) en su cara.

DATASET

ANGRY

angry

2256 files

RELAXED

relaxed

4349 files

HAPPY

happy

4784 files

SAD

sad

4532 files

MODELOS DE CLASIFICASIÓN

GAUSSIAN NB

```python
from sklearn.model_selection import cross_val_score, KFold
from sklearn.metrics import make_scorer, accuracy_score, classification_report

from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import cross_val_score, KFold
from sklearn.metrics import make_scorer, accuracy_score, classification_report
import time

inicio = time.time()

# Aplanar las imágenes
X_train_flattened = X_train.reshape(X_train.shape[0], -1)
X_test_flattened = X_test.reshape(X_test.shape[0], -1)

estimador = GaussianNB()
estimador.fit(X_train_flattened, y_train)
y_pred = estimador.predict(X_test_flattened)

score = cross_val_score(estimador, X_train_flattened, y_train, cv=KFold(10, shuffle=True), scoring=make_scorer(accuracy_score))

fin = time.time()

print("Tiempo de ejecución: %.3f segundos" % (fin - inicio))
print("Accuracy Score: %.3f (+/- %.5f)" % (np.mean(score), np.std(score)))
print(classification_report(y_test, y_pred))
```

```
Tiempo de ejecución: 119.057 segundos
Accuracy Score: 0.310 (+/- 0.01293)
              precision    recall  f1-score   support

       angry       0.17      0.19      0.18       469
       happy       0.32      0.48      0.39       926
     relaxed       0.31      0.13      0.18       916
         sad       0.33      0.34      0.34       874

    accuracy                           0.30      3185
   macro avg       0.28      0.29      0.27      3185
weighted avg       0.30      0.30      0.28      3185
```

MODELOS DE CLASIFICASIÓN

RANDOM FOREST CLASSIFIER

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score, KFold
from sklearn.metrics import make_scorer, accuracy_score, classification_report
import time

inicio = time.time()

# Aplanar las imágenes
X_train_flattened = X_train.reshape(X_train.shape[0], -1)
X_test_flattened = X_test.reshape(X_test.shape[0], -1)

# Crear una instancia del clasificador Random Forest
clf = RandomForestClassifier(n_estimators=100, random_state=42)

# Entrenar el clasificador
clf.fit(X_train_flattened, y_train)

# Realizar predicciones en los datos de prueba
y_pred = clf.predict(X_test_flattened)

# Calcular el puntaje de validación cruzada
score = cross_val_score(clf, X_train_flattened, y_train, cv=KFold(10, shuffle=True), scoring=make_scorer(accuracy_score))

fin = time.time()

print("Tiempo de ejecución: %.3f segundos" % (fin - inicio))
print("Accuracy Score: %.3f (+/- %.5f)" % (np.mean(score), np.std(score)))
print(classification_report(y_test, y_pred))
```

```
Tiempo de ejecución: 3033.728 segundos
Accuracy Score: 0.357 (+/- 0.00953)
              precision    recall  f1-score   support

       angry       0.30      0.01      0.03       456
       happy       0.38      0.52      0.44       963
     relaxed       0.34      0.29      0.32       889
         sad       0.34      0.43      0.38       877

    accuracy                           0.36      3185
   macro avg       0.34      0.31      0.29      3185
weighted avg       0.35      0.36      0.33      3185
```

MODELOS DE CLASIFICASIÓN

DECISION TREE CLASSIFIER

```python
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score, KFold
from sklearn.metrics import make_scorer, accuracy_score, classification_report
import time

inicio = time.time()

# Aplanar las imágenes
X_train_flattened = X_train.reshape(X_train.shape[0], -1)
X_test_flattened = X_test.reshape(X_test.shape[0], -1)

# Crear una instancia del clasificador Decision Tree
clf = DecisionTreeClassifier(random_state=42)

# Entrenar el clasificador
clf.fit(X_train_flattened, y_train)

# Realizar predicciones en los datos de prueba
y_pred = clf.predict(X_test_flattened)

# Calcular el puntaje de validación cruzada
score = cross_val_score(clf, X_train_flattened, y_train, cv=KFold(10, shuffle=True), scoring=make_scorer(accuracy_score))

fin = time.time()

print("Tiempo de ejecución: %.3f segundos" % (fin - inicio))
print("Accuracy Score: %.3f (+/- %.5f)" % (np.mean(score), np.std(score)))
print(classification_report(y_test, y_pred))
```

```
Tiempo de ejecución: 9645.359 segundos
Accuracy Score: 0.284 (+/- 0.01237)
              precision    recall  f1-score   support

       angry       0.17      0.17      0.17       456
       happy       0.33      0.32      0.32       963
     relaxed       0.29      0.29      0.29       889
         sad       0.31      0.32      0.31       877

    accuracy                           0.29      3185
   macro avg       0.27      0.27      0.27      3185
weighted avg       0.29      0.29      0.29      3185
```

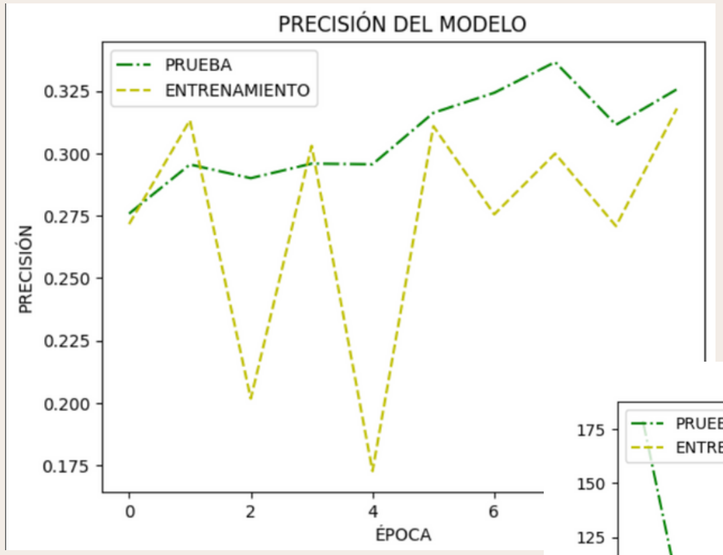
MODELOS DE CLASIFICASIÓN

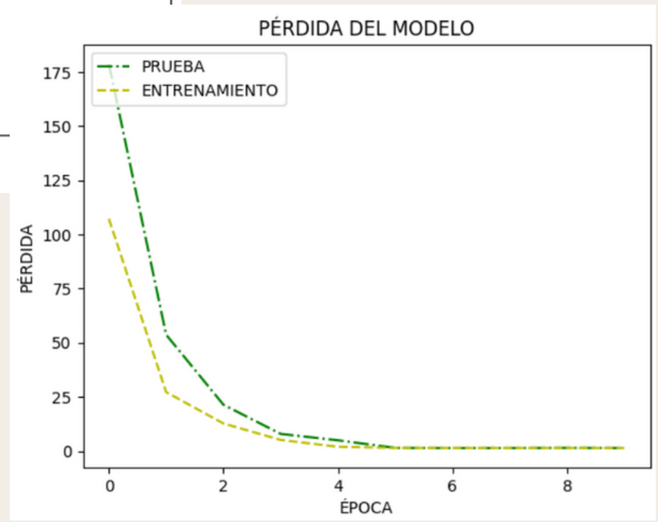
IMPLEMENTACIÓN DE SVC

```python
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report

# Aplanar las imágenes
X_train_flattened = X_train.reshape(X_train.shape[0], -1)
X_test_flattened = X_test.reshape(X_test.shape[0], -1)

# Crear una instancia del clasificador SVM
clf = SVC()

# Entrenar el clasificador
clf.fit(X_train_flattened, y_train)

# Realizar predicciones en los datos de prueba
y_pred = clf.predict(X_test_flattened)

# Calcular la precisión del clasificador
accuracy = accuracy_score(y_test, y_pred)

# Imprimir la precisión y el informe de clasificación
print("Precisión del clasificador SVM: %.3f" % accuracy)
print(classification_report(y_test, y_pred))
```

```
Precisión del clasificador SVM: 0.363
              precision    recall  f1-score   support

       angry       0.00      0.00      0.00       484
       happy       0.38      0.55      0.45       965
     relaxed       0.35      0.30      0.32       854
         sad       0.34      0.42      0.38       882

    accuracy                           0.36      3185
   macro avg       0.27      0.32      0.29      3185
weighted avg       0.31      0.36      0.33      3185
```

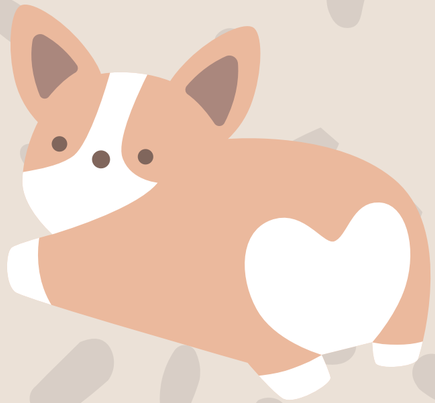
RED NEURONAL





A FUTURO

A futuro se espera poder crear un aplicativo que permita subir fotos o en su defecto capturar en tiempo real y que se pueda usar la IA.



¡GRACIAS!