K-NN Αλγόριθμος

Για τον kNN αρχικά κάναμε import τις βιβλιοθήκες που θα χρησιμοποιήσουμε. Στην συνέχεια διαβάζουμε τα αρχεία με τα data και ξεκινάμε την επεξεργασία τους. Φορτώνω στην μεταβλητή Y τον τύπο(type) από το test.csv και στην συνέχεια αφαιρώ από από την μεταβλητή train_data το id και type και στο test_data το id (το type δεν υπάρχει σε αυτό το αρχείο). Οι εντολές φαίνοντε στην εικόνα από κάτω.

```
Y = train_data['type']
train_data = train_data.drop(['id'], axis=1)
train_data = train_data.drop(['type'], axis=1)
test_data = test_data.drop(['id'], axis=1)
```

Στην συνέχεια μετατρέπω στο train_data και test_data το χρώμα από sting σε int, από 1 έως 6. Επειδή τα υπόλοιπα δεδομένα είναι δεκαδικά για να μην έχουν μεγάλη απόκληση τα διαιρέσαμε με το 6 έτσι ώστε να πέρνουν τιμέ έως 1. Οι εντολές φαίνοντε από κάτω.

```
train_data['color'] = train_data['color'].replace({'clear': str(1/6)})
train_data['color'] = train_data['color'].replace({'green': str(2/6)})
train_data['color'] = train_data['color'].replace({'black': str(3/6)})
train_data['color'] = train_data['color'].replace({'blue': str(4/6)})
train_data['color'] = train_data['color'].replace({'white': str(5/6)})
train_data['color'] = train_data['color'].replace({'blood': str(6/6)})
```

```
test_data['color'] = test_data['color'].replace({'clear': str(1/6)})
test_data['color'] = test_data['color'].replace({'green': str(2/6)})
test_data['color'] = test_data['color'].replace({'black': str(3/6)})
test_data['color'] = test_data['color'].replace({'blue': str(4/6)})
test_data['color'] = test_data['color'].replace({'white': str(5/6)})
test_data['color'] = test_data['color'].replace({'blood': str(6/6)})
```

Για τις τιμές των φαντασμάτων δώσαμε από 0 έως 2, όπως φαίνεται.

```
Y = [0 if y == 'Ghoul' else 1 if y == 'Goblin' else 2 for y in Y]
```

Στην συνέχεια για να υπολογίσουμε τον f1_score κάναμε split τα δεδομένα του train_data, ορίσαμε τον αλγόριθμο, τον εκπαιδεύσαμε και κάναμε την πρόβλεψη και τέλος εκτυπώσαμε το f1_score. Ο Κώδικας φαίνετε παρακάτω.

```
X_train, X_test, y_train, y_test = train_test_split(train_data, Y, random_state=0, test_size=0.2)


classifier = KNeighborsClassifier(n_neighbors=10, metric='euclidean')
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

print("f1 score: ", metrics.f1_score(y_test, y_pred, average='weighted'))
```

Για να πάρουμε το accuracy από το kaggle ακολουθήσαμε την ίδια τακτική . Αλλά εκπαιδεύσαμε τον αλγόριθμο με τα train_data,Y και κάναμε την πρόβλεψη με το test_data, Στην συνέχεια μετατρέψαμε τα 0,1 και 2 που βγήκαν από την πρόβλεψη στα αντίστοιχα (Ghoul, Goblin, Ghost) και τα περάσαμε σε ένα αρχείο knn_submission.csv όπου δώσαμε στο kaggle και μας έδωσε τα απότελέσματα.

```
knn = KNeighborsClassifier(n_neighbors = 10)

knn.fit(train_data, Y)

res = knn.predict(test_data)

type = ['Ghoul' if r == 0 else 'Goblin' if r == 1 else 'Ghost' for r in res]
type = pd.Series(type)

id = sample['id']

df = pd.DataFrame({'type': type})
df['id'] = id
#df.columns = ['ImageId', 'Label']
df = df[['id', 'type']]
df.columns
print(df.head(5))
#df['type'].value_counts()


df.to_csv('knn_submission.csv', index=False)
```

Τα αποτελέσματα για k=1, k=3, k=5, k=10

Για k=1: f1 score: 0.7764138108073418
Score: 0.67674

Για k=3: f1 score: 0.7767682714321715
Score: 0.66918

Για k=5: f1 score: 0.7603496210288663
Score: 0.70510

Για k=10: f1 score: 0.772880174291939
Score: 0.70699

SVM Αλγόριθμος

Για τον SVM επεξεργαστήκαμε με τον ίδιο τρόπο τα δεδομένα όπως και για τον k-NN. Και με το ίδιο σκεπτικό βρήκαμε το f1_score κάνοντας split τα δεδομένα απο το train.csv. Παρακάτω φαίνετε ο κώδικας που αρχικοποιούμε τον αλγόριθμο τον εκπαιδεύουμε, κάνουμε την πρόβλεψη και τυπώνουμε το f1 καθώς και πως περνάμε τα αρχεία στο svm_submission.csv για να το δώσουμε στο kaggle να πάρουμε το score.

```
X_train, X_test, y_train, y_test = train_test_split(train_data, Y, random_state=0, test_size=0.2)

#svm = SVC(kernel="linear")
svm = SVC(kernel="rbf") # gaussian

svm.fit(X_train, y_train)
y_pred = svm.predict(X_test)
print("f1 score: ", metrics.f1_score(y_test, y_pred, average='weighted'))

################################################################################

#svm = SVC(kernel="linear")
svm = SVC(kernel="rbf") # gaussian
svm.fit(train_data, Y)
res = svm.predict(test_data)

type = ['Ghoul' if r == 0 else 'Goblin' if r == 1 else 'Ghost' for r in res]
type = pd.Series(type)


id = sample['id']

df = pd.DataFrame({'type': type})
df['id'] = id
df = df[['id', 'type']]
df.columns
print(df.head(5))

df.to_csv('svm_submission.csv', index=False)
```

Τα αποτελέσματα για γραμική(linear kernel) και Gaussian(RBF kernel)

Για linear kernel: f1 score: 0.7678659944706457
Score: 0.73913
Για RBF kernel: f1 score: 0.7998639455782314
Score: 0.72211

Neural Networks αλγόριθμος

Παρόμοια φορτώσαμε τα δεδομένα train.csv, data.csv και ξεκινήσαμε την διαμόρφωση τους.

Χρησιμοποιόντας τις δύο παρακάτω εντολές διαμορφώσαμε τις τιμές για τα χρώματα.

```
train_data = pd.concat([train_data, pd.get_dummies(train_data['color'])], axis=1)
train_data = train_data.drop('color', axis=1)
```

Με όμοιο τρόπο διαμορφώσαμε το χρώμα και για το test_data.

Στην συνέχεια κάναμε split τα δεδομένα όπως και στους παραπάνω αλγορίθμους και ορίσαμε τον αλγόριθμο, κάναμε compile και τον εκπαιδέυσαμε και κάναμε την πρόβλεψη για να πάρουμε το score.

```
model = Sequential()
model.add(Dense(64, input_shape=(X_train.shape[1],)))
model.add(Dense(50, activation='sigmoid',))
#model.add(Dense(50, activation='sigmoid'))
model.add(Dense(3, activation='softmax'))
model.summary()

model.compile(optimizer='sgd',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model_data = model.fit(X_train, Y_train,
          validation_data=(X_test, Y_test),
          verbose=2,
          epochs=10,
          batch_size=16)

pred=model.predict(test_data.drop('id',axis=1))
```

Τα αποτελέσματα:
Για ένα κρυμμένο επίπεδο (50): Score: 0.34026

Για ένα κρυμμένο επίπεδο (100): Score: 0.40264

Για ένα κρυμμένο επίπεδο (200): Score: 0.34593

Για δύο κρυμμένα επίπεδα (50, 25): Score: 0.31001

Για δύο κρυμμένα επίπεδα (100, 50): Score: 0.32325

Για δύο κρυμμένα επίπεδα (200, 100): Score: 0.41776