model_selection_training

April 26, 2024

[ ]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
```

[ ]:

```
data = pd.read_csv('./final_data/final_data.csv')
```

[ ]:

```
# Split the dataset into features (X) and target variable (y)
X = data.drop(columns=['compound'])
y = data['compound']
```

[ ]:

```
# Encode categorical variables using one-hot encoding
X_encoded = pd.get_dummies(X)
```

[ ]:

```
#Split the dataset into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.
  ↪2, random_state=42)
```

[ ]:

```
# Make predictions on new data
# new data is the first row of the original data
new_data = data.iloc[[0]]
new_data = new_data.drop(columns=["compound"])

# Encode new data using the same encoding as the training data
new_data_encoded = pd.get_dummies(new_data)
```

Random Forest Classifier

[ ]:

```
# Choose a machine learning algorithm
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
model.fit(X_train, y_train)
```

[ ]:

```
RandomForestClassifier(random_state=42)
```

[ ]:

```
# Evaluate the model
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')

# Print classification report
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.7356521739130435
              precision    recall  f1-score   support

        HARD       0.77      0.80      0.78       271
   HYPERSOFT       0.71      0.80      0.75        15
INTERMEDIATE       0.91      0.97      0.94        64
      MEDIUM       0.73      0.68      0.70       391
        SOFT       0.70      0.75      0.72       293
   SUPERSOFT       0.71      0.64      0.67        53
   ULTRASOFT       0.60      0.57      0.58        49
         WET       0.90      0.64      0.75        14

    accuracy                           0.74      1150
   macro avg       0.75      0.73      0.74      1150
weighted avg       0.74      0.74      0.73      1150
```

[ ]:

```
# Make predictions
predictions = model.predict(new_data_encoded)
print(f'Predicted tire compound: {predictions}')
```

```
Predicted tire compound: ['ULTRASOFT']
```

Gradient Boosting Classifier

[ ]:

```
# Choose a machine learning algorithm
model = GradientBoostingClassifier(n_estimators=100, random_state=42)

# Train the model
model.fit(X_train, y_train)
```

[ ]:

```
GradientBoostingClassifier(random_state=42)
```

[ ]:

```
# Evaluate the model
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')

# Print classification report
```

```python
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.7365217391304347
              precision    recall  f1-score   support

        HARD       0.75      0.77      0.76       271
   HYPERSOFT       0.71      0.67      0.69        15
INTERMEDIATE       0.93      0.97      0.95        64
      MEDIUM       0.70      0.71      0.70       391
        SOFT       0.73      0.74      0.74       293
   SUPERSOFT       0.73      0.68      0.71        53
   ULTRASOFT       0.66      0.59      0.62        49
         WET       0.90      0.64      0.75        14

    accuracy                           0.74      1150
   macro avg       0.76      0.72      0.74      1150
weighted avg       0.74      0.74      0.74      1150
```

```python
[ ]: # Make predictions
predictions = model.predict(new_data_encoded)
print(f'Predicted tire compound: {predictions}')
```

```
Predicted tire compound: ['ULTRASOFT']
```

Support Vector Machines with RBF kernel

```python
[ ]: # Choose a machine learning algorithm
model = SVC(kernel='rbf', random_state=42)

# Train the model
model.fit(X_train, y_train)
```

```
[ ]: SVC(random_state=42)
```

```python
[ ]: # Evaluate the model
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')

# Print classification report
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.3391304347826087
              precision    recall  f1-score   support

        HARD       0.00      0.00      0.00       271
   HYPERSOFT       0.00      0.00      0.00        15
INTERMEDIATE       0.00      0.00      0.00        64
```

```
      MEDIUM       0.34      1.00      0.51       391
        SOFT       0.00      0.00      0.00       293
   SUPERSOFT       0.00      0.00      0.00        53
   ULTRASOFT       0.00      0.00      0.00        49
     UNKNOWN       0.00      0.00      0.00         0
         WET       0.00      0.00      0.00        14

    accuracy                           0.34      1150
   macro avg       0.04      0.11      0.06      1150
weighted avg       0.12      0.34      0.17      1150
```

```
/Users/neahabijo/anaconda3/envs/tyre_pred/lib/python3.12/site-
packages/sklearn/metrics/_classification.py:1509: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/neahabijo/anaconda3/envs/tyre_pred/lib/python3.12/site-
packages/sklearn/metrics/_classification.py:1509: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 in labels with no true samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/neahabijo/anaconda3/envs/tyre_pred/lib/python3.12/site-
packages/sklearn/metrics/_classification.py:1509: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/neahabijo/anaconda3/envs/tyre_pred/lib/python3.12/site-
packages/sklearn/metrics/_classification.py:1509: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 in labels with no true samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/neahabijo/anaconda3/envs/tyre_pred/lib/python3.12/site-
packages/sklearn/metrics/_classification.py:1509: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/neahabijo/anaconda3/envs/tyre_pred/lib/python3.12/site-
packages/sklearn/metrics/_classification.py:1509: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 in labels with no true samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

[ ]:

```python
# Make predictions
predictions = model.predict(new_data_encoded)
print(f'Predicted tire compound: {predictions}')
```

```
Predicted tire compound: ['MEDIUM']
```

K Nearest Neighbours Classifiers

```
[ ]: model = KNeighborsClassifier(n_neighbors=5)

     # Train the model
     model.fit(X_train, y_train)
```

```
[ ]: KNeighborsClassifier()
```

```
[ ]: # Evaluate the model
     y_pred = model.predict(X_test)
     accuracy = accuracy_score(y_test, y_pred)
     print(f'Accuracy: {accuracy}')

     # Print classification report
     print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.3243478260869565
              precision    recall  f1-score   support

        HARD       0.29      0.39      0.33       271
   HYPERSOFT       0.30      0.20      0.24        15
INTERMEDIATE       0.42      0.55      0.47        64
      MEDIUM       0.34      0.34      0.34       391
        SOFT       0.35      0.29      0.32       293
   SUPERSOFT       0.33      0.11      0.17        53
   ULTRASOFT       0.12      0.06      0.08        49
     UNKNOWN       0.00      0.00      0.00         0
         WET       1.00      0.07      0.13        14

    accuracy                           0.32      1150
   macro avg       0.35      0.22      0.23      1150
weighted avg       0.33      0.32      0.32      1150


/Users/neahabijo/anaconda3/envs/tyre_pred/lib/python3.12/site-
packages/sklearn/metrics/_classification.py:1509: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 in labels with no true samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/neahabijo/anaconda3/envs/tyre_pred/lib/python3.12/site-
packages/sklearn/metrics/_classification.py:1509: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 in labels with no true samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/neahabijo/anaconda3/envs/tyre_pred/lib/python3.12/site-
packages/sklearn/metrics/_classification.py:1509: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 in labels with no true samples. Use
`zero_division` parameter to control this behavior.
```

```
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

[ ]:

```python
# Make predictions
predictions = model.predict(new_data_encoded)
print(f'Predicted tire compound: {predictions}')
```

```
Predicted tire compound: ['HARD']
```

Multi-Layer Perceptron Classifier

[ ]:

```python
model = MLPClassifier(hidden_layer_sizes=(100, 50), max_iter=500,␣
 ↪random_state=42)

# Train the model
model.fit(X_train, y_train)
```

[ ]:

```
MLPClassifier(hidden_layer_sizes=(100, 50), max_iter=500, random_state=42)
```

[ ]:

```python
# Evaluate the model
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')

# Print classification report
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.23565217391304347
              precision    recall  f1-score   support

        HARD       0.24      1.00      0.38       271
   HYPERSOFT       0.00      0.00      0.00        15
INTERMEDIATE       0.00      0.00      0.00        64
      MEDIUM       0.00      0.00      0.00       391
        SOFT       0.00      0.00      0.00       293
   SUPERSOFT       0.00      0.00      0.00        53
   ULTRASOFT       0.00      0.00      0.00        49
         WET       0.00      0.00      0.00        14

    accuracy                           0.24      1150
   macro avg       0.03      0.12      0.05      1150
weighted avg       0.06      0.24      0.09      1150

/Users/neahabijo/anaconda3/envs/tyre_pred/lib/python3.12/site-
packages/sklearn/metrics/_classification.py:1509: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/neahabijo/anaconda3/envs/tyre_pred/lib/python3.12/site-
packages/sklearn/metrics/_classification.py:1509: UndefinedMetricWarning:
```

```
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/neahabijo/anaconda3/envs/tyre_pred/lib/python3.12/site-
packages/sklearn/metrics/_classification.py:1509: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

[ ]:

```
# Make predictions
predictions = model.predict(new_data_encoded)
print(f'Predicted tire compound: {predictions}')
```

```
Predicted tire compound: ['HARD']
```

Decision Tree Classifier

[ ]:

```
model = DecisionTreeClassifier(random_state=42)

# Train the model
model.fit(X_train, y_train)
```

[ ]:

```
DecisionTreeClassifier(random_state=42)
```

[ ]:

```
# Evaluate the model
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')

# Print classification report
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.6843478260869565
              precision    recall  f1-score   support

        HARD       0.74      0.70      0.72       271
   HYPERSOFT       0.65      0.87      0.74        15
INTERMEDIATE       0.91      0.94      0.92        64
      MEDIUM       0.64      0.63      0.63       391
        SOFT       0.67      0.71      0.69       293
   SUPERSOFT       0.60      0.64      0.62        53
   ULTRASOFT       0.56      0.47      0.51        49
         WET       0.91      0.71      0.80        14

    accuracy                           0.68      1150
   macro avg       0.71      0.71      0.71      1150
weighted avg       0.68      0.68      0.68      1150
```

[ ]:

```python
# Make predictions
predictions = model.predict(new_data_encoded)
print(f'Predicted tire compound: {predictions}')
```

```
Predicted tire compound: ['ULTRASOFT']
```