cameranotebook

April 14, 2025

[4]:
```
#cameras.csv is a dataset that I found it it had to do with different cameras
#it was about the prices and being able to predict where the camera lands in␣
 ↪the price range


#logistic regression was a linear model
#k-NN used 5 neighbors and went for the distance based approach
#Decision tree splits into future thresholds


#Each model has their strengths and weaknesses.

#Linear regression had an accuraacy of 59% for all predictions, but for high it␣
 ↪had
#a 84%. On recall its was 86% accurate for low. On F1-score low was .69 since␣
 ↪it had high recalls
#It struggles with medium recalls. Accuracy is low overall with 59%

#k-NN had an accuracy of 63% for all predictions. It had a 68% precision for␣
 ↪low.
#Recalls 78% on lows as well and a .73 on low F1-score score, making this model␣
 ↪strong for low predictions
#weak on high predictions

#Decision tree had an accuracy of 67% for all predictions. It had a 74% for low␣
 ↪precision
#73% for low recall and a .73 for F1- score. This is good with low and mediums␣
 ↪overall best one.
```

[5]:
```
import pandas as pd

# Load the data
df = pd.read_csv("cameras.csv")

# Create a target column with categories
def price_category(price):
    if price < 200:
```

```
        return 'Low'
    elif price <= 600:
        return 'Medium'
    else:
        return 'High'

# Applying the functions to create new column
df['PriceCategory'] = df['Price'].apply(price_category)


## Cleaning data
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder

# Drop columns that are not needed
df_clean = df.drop(columns=['Model', 'Price', 'Macro focus range', 'Storage␣
 ↪included', 'Dimensions'])

# Drop rows without values
df_clean = df_clean.dropna()

X = pd.get_dummies(df_clean.drop(columns='PriceCategory'), drop_first=True)

# Encode the target variable
le = LabelEncoder()
y = le.fit_transform(df_clean['PriceCategory'])

# Split it
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Standardize the feature values
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)


#Classification Models
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier

#Initialize models
log_reg = LogisticRegression(max_iter=1000)
knn = KNeighborsClassifier(n_neighbors=5)
tree = DecisionTreeClassifier(random_state=42)
```

```python
log_reg.fit(X_train_scaled, y_train)
knn.fit(X_train_scaled, y_train)
tree.fit(X_train, y_train)

#Evaluate the Models
from sklearn.metrics import classification_report

# Make predictions
y_pred_log = log_reg.predict(X_test_scaled)
y_pred_knn = knn.predict(X_test_scaled)
y_pred_tree = tree.predict(X_test)

print("Logistic Regression:\n")
print(classification_report(y_test, y_pred_log, target_names=le.classes_))

print("k-NN:\n")
print(classification_report(y_test, y_pred_knn, target_names=le.classes_))

print("Decision Tree:\n")
print(classification_report(y_test, y_pred_tree, target_names=le.classes_))
```

```
Logistic Regression:

              precision    recall  f1-score   support

        High       0.84      0.47      0.60        34
         Low       0.57      0.86      0.69        99
      Medium       0.55      0.29      0.38        75

    accuracy                           0.59       208
   macro avg       0.65      0.54      0.56       208
weighted avg       0.61      0.59      0.56       208

k-NN:

              precision    recall  f1-score   support

        High       0.49      0.56      0.52        34
         Low       0.68      0.78      0.73        99
      Medium       0.64      0.48      0.55        75

    accuracy                           0.63       208
   macro avg       0.60      0.61      0.60       208
weighted avg       0.64      0.63      0.63       208

Decision Tree:
```

```
              precision    recall  f1-score   support

        High       0.63      0.56      0.59        34
         Low       0.74      0.73      0.73        99
      Medium       0.59      0.64      0.62        75

    accuracy                           0.67       208
   macro avg       0.66      0.64      0.65       208
weighted avg       0.67      0.67      0.67       208
```

[ ]: