final

March 14, 2025

1 Importing Required Libraries

[1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score, confusion_matrix
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.preprocessing import StandardScaler
```

2 Loading and Preprocessing Data

[2]:
```
df = pd.read_csv('data/magic04.data')

X = df.iloc[:, :-1]  # Features
y = df.iloc[:, -1]   # Target

# Split the data into train, temp (val+test)
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)

# Split temp into validation and test
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5,␣
 ↪random_state=42)

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)
X_test_scaled = scaler.transform(X_test)

# Confirming everything is okay (size)
print(f'X_train_scaled shape: {X_train_scaled.shape}')
print(f'X_val_scaled shape: {X_val_scaled.shape}')
```

```
print(f'X_test_scaled shape: {X_test_scaled.shape}')
```

```
X_train_scaled shape: (13313, 10)
X_val_scaled shape: (2853, 10)
X_test_scaled shape: (2853, 10)
```

3 K-Nearest Neighbors Classification

[3]:
```
k_values = [3, 5, 7, 9]
classification_results = {}

for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_val)

    classification_results[k] = {
        'accuracy': accuracy_score(y_val, y_pred),
        'precision': precision_score(y_val, y_pred, pos_label='g',␣
 ↪zero_division=0),
        'recall': recall_score(y_val, y_pred, pos_label='g', zero_division=0),
        'f1_score': f1_score(y_val, y_pred, pos_label='g', zero_division=0),
        'confusion_matrix': confusion_matrix(y_val, y_pred)
    }

# Display results
for k, metrics in classification_results.items():
    print(f'Results for k = {k}')
    for metric, value in metrics.items():
        print(f'{metric}: {value}')
```

```
Results for k = 3
accuracy: 0.8002103049421662
precision: 0.8091127098321343
recall: 0.907477138246369
f1_score: 0.8554766734279919
confusion_matrix: [[1687  172]
 [ 398  596]]
Results for k = 5
accuracy: 0.804416403785489
precision: 0.8055425082198215
recall: 0.9225389994620764
f1_score: 0.8600802407221665
confusion_matrix: [[1715  144]
 [ 414  580]]
Results for k = 7
accuracy: 0.8138801261829653
```

```
precision: 0.807123034227567
recall: 0.9386767079074771
f1_score: 0.8679432976871425
confusion_matrix: [[1745  114]
 [ 417  577]]
Results for k = 9
accuracy: 0.8152821591307395
precision: 0.8057851239669421
recall: 0.9440559440559441
f1_score: 0.8694575179588804
confusion_matrix: [[1755  104]
 [ 423  571]]
```

4 Regression Models: Linear, Lasso, and Ridge

[4]:
```
df = pd.read_csv('data/California_Houses.csv')

X = df.iloc[:, 1:]  # Features
y = df.iloc[:, 0].to_frame()   # Target (House Value)

# Split the data into train, temp (val+test)
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)

# Split temp into validation and test
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5,␣
 ↪random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_val = scaler.transform(X_val)
X_test = scaler.transform(X_test)

scaler_y = StandardScaler()
y_train = scaler_y.fit_transform(y_train)  # Fit on train only
y_val = scaler_y.transform(y_val)  # Transform val
y_test = scaler_y.transform(y_test)  # Transform test

# Linear Regression
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
y_pred_lr = lr_model.predict(X_val)
mse_lr = mean_squared_error(y_val, y_pred_lr)
mae_lr = mean_absolute_error(y_val, y_pred_lr)

# Lasso Regression
```

```
lasso_model = Lasso(alpha=1.0)
lasso_model.fit(X_train, y_train)
y_pred_lasso = lasso_model.predict(X_val)
mse_lasso = mean_squared_error(y_val, y_pred_lasso)
mae_lasso = mean_absolute_error(y_val, y_pred_lasso)

# Ridge Regression
ridge_model = Ridge(alpha=1.0)
ridge_model.fit(X_train, y_train)
y_pred_ridge = ridge_model.predict(X_val)
mse_ridge = mean_squared_error(y_val, y_pred_ridge)
mae_ridge = mean_absolute_error(y_val, y_pred_ridge)

# Display results
print(f'Linear Regression: MSE = {mse_lr}, MAE = {mae_lr}')
print(f'Lasso Regression: MSE = {mse_lasso}, MAE = {mae_lasso}')
print(f'Ridge Regression: MSE = {mse_ridge}, MAE = {mae_ridge}')
```

```
Linear Regression: MSE = 0.3662929560059561, MAE = 0.4388090758856613
Lasso Regression: MSE = 0.9724365787799097, MAE = 0.7781121060057162
Ridge Regression: MSE = 0.3662981103149209, MAE = 0.43883974666966097
```