Linear Regression

Steps:

1. Import necessary libraries

2. Load and preprocess the dataset

3. Split the data into training and testing sets

4. Train the Linear Regression model

5. Predict using the test set

6. Evaluate the model

Code:

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import pandas as pd

# Load data
data = pd.read_csv('your_data.csv')
X = data[['feature1', 'feature2']]
y = data['target']

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

# Train model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# Evaluate
print('MSE:', mean_squared_error(y_test, y_pred))
```

Logistic Regression

Steps:

1. Import necessary libraries

2. Load and preprocess the dataset

3. Split the data into training and testing sets

4. Train the Logistic Regression model

5. Predict using the test set

6. Evaluate the model using classification metrics

Code:

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import pandas as pd

# Load data
data = pd.read_csv('your_data.csv')
X = data[['feature1', 'feature2']]
y = data['target']

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

# Train model
model = LogisticRegression()
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# Evaluate
print(classification_report(y_test, y_pred))
```

Polynomial Regression

Steps:

1. Import necessary libraries

2. Load and preprocess the dataset

3. Transform features into polynomial features

4. Split the data into training and testing sets

5. Train the Linear Regression model

6. Predict using the test set

7. Evaluate the model

Code:

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import pandas as pd

# Load data
data = pd.read_csv('your_data.csv')
X = data[['feature1']]
y = data['target']

# Polynomial features
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)

# Split data
X_train, X_test, y_train, y_test = train_test_split(X_poly, y, test_size=0.2,
random_state=42)

# Train model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# Evaluate
print('MSE:', mean_squared_error(y_test, y_pred))
```

Lasso Regression

Steps:

1. Import necessary libraries

2. Load and preprocess the dataset

3. Split the data into training and testing sets

4. Train the Lasso Regression model

5. Predict using the test set

6. Evaluate the model

Code:

```
from sklearn.linear_model import Lasso
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import pandas as pd

# Load data
data = pd.read_csv('your_data.csv')
X = data[['feature1', 'feature2']]
y = data['target']

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

# Train model
model = Lasso(alpha=0.1)
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# Evaluate
print('MSE:', mean_squared_error(y_test, y_pred))
```

Ridge Regression

Steps:

1. Import necessary libraries

2. Load and preprocess the dataset

3. Split the data into training and testing sets

4. Train the Ridge Regression model

5. Predict using the test set

6. Evaluate the model

Code:

```
from sklearn.linear_model import Ridge
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import pandas as pd

# Load data
data = pd.read_csv('your_data.csv')
X = data[['feature1', 'feature2']]
y = data['target']

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

# Train model
model = Ridge(alpha=1.0)
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# Evaluate
print('MSE:', mean_squared_error(y_test, y_pred))
```

Decision Tree Regression

Steps:

1. Import necessary libraries

2. Load and preprocess the dataset

3. Split the data into training and testing sets

4. Train the Decision Tree model

5. Predict using the test set

6. Evaluate the model

Code:

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import pandas as pd

# Load data
data = pd.read_csv('your_data.csv')
X = data[['feature1', 'feature2']]
y = data['target']

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

# Train model
model = DecisionTreeRegressor()
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# Evaluate
print('MSE:', mean_squared_error(y_test, y_pred))
```

Random Forest Regression

Steps:

1. Import necessary libraries

2. Load and preprocess the dataset

3. Split the data into training and testing sets

4. Train the Random Forest model

5. Predict using the test set

6. Evaluate the model

Code:

```
from sklearn.ensemble import RandomForestRegressor
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import pandas as pd

# Load data
data = pd.read_csv('your_data.csv')
X = data[['feature1', 'feature2']]
y = data['target']

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

# Train model
model = RandomForestRegressor(n_estimators=100)
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# Evaluate
print('MSE:', mean_squared_error(y_test, y_pred))
```