Scikit Library

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder, OneHotEncoder
from sklearn.model_selection import train_test_split
import numpy as np

# Sample data
X = np.array([[1, 2], [3, 4], [5, 6]])
y = np.array([0, 1, 0])
labels = np.array(['cat', 'dog', 'cat'])

# 1. StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 2. MinMaxScaler
minmax = MinMaxScaler()
X_minmax = minmax.fit_transform(X)

# 3. LabelEncoder
le = LabelEncoder()
y_encoded = le.fit_transform(labels)

# 4. OneHotEncoder
ohe = OneHotEncoder(sparse=False)
X_ohe = ohe.fit_transform(np.array(labels).reshape(-1, 1))

# 5. train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)

from sklearn.model_selection import cross_val_score, GridSearchCV
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.linear_model import LogisticRegression

# Classifier for demonstration
model = LogisticRegression()

# 6. cross_val_score
scores = cross_val_score(model, X, y, cv=3)

# 7. GridSearchCV
param_grid = {'C': [0.1, 1, 10]}
grid = GridSearchCV(LogisticRegression(), param_grid, cv=3)
grid.fit(X, y)

# Fit model
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

# 8. accuracy_score
accuracy = accuracy_score(y_test, y_pred)

# 9. confusion_matrix
cm = confusion_matrix(y_test, y_pred)

# 10. classification_report
report = classification_report(y_test, y_pred)

from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

# 11. LogisticRegression
lr = LogisticRegression().fit(X_train, y_train)

# 12. GaussianNB
gnb = GaussianNB().fit(X_train, y_train)

# 13. SVC
svc = SVC().fit(X_train, y_train)

# 14. DecisionTreeClassifier
dt = DecisionTreeClassifier().fit(X_train, y_train)

# 15. RandomForestClassifier
rfc = RandomForestClassifier().fit(X_train, y_train)


from sklearn.linear_model import LinearRegression, Ridge
from sklearn.ensemble import RandomForestRegressor

# Sample regression data
Xr = np.array([[1], [2], [3], [4], [5]])
yr = np.array([1, 4, 9, 16, 25])

# 16. LinearRegression
lin_reg = LinearRegression().fit(Xr, yr)

# 17. Ridge Regression
ridge = Ridge(alpha=1.0).fit(Xr, yr)

# 18. RandomForestRegressor
rf_reg = RandomForestRegressor().fit(Xr, yr)
```

```
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA

# 19. KMeans
kmeans = KMeans(n_clusters=2, random_state=0).fit(X)

# 20. PCA
pca = PCA(n_components=1)
X_pca = pca.fit_transform(X)
```

seaborn Library

```
import seaborn as sns
import matplotlib.pyplot as plt

# Load built-in dataset
df = sns.load_dataset("tips")
# 1. sns.scatterplot
sns.scatterplot(x="total_bill", y="tip", data=df)
plt.show()

# 2. sns.lineplot
sns.lineplot(x="size", y="tip", data=df)
plt.show()

# 3. sns.histplot
sns.histplot(df["total_bill"], kde=True)
plt.show()

# 4. sns.boxplot
sns.boxplot(x="day", y="total_bill", data=df)
plt.show()

# 5. sns.violinplot
sns.violinplot(x="day", y="tip", data=df)
plt.show()

# 6. sns.barplot
sns.barplot(x="sex", y="tip", data=df)
plt.show()

# 7. sns.countplot
sns.countplot(x="day", data=df)
plt.show()

# 8. sns.stripplot
sns.stripplot(x="day", y="tip", data=df, jitter=True)
plt.show()

# 9. sns.swarmplot
sns.swarmplot(x="day", y="tip", data=df)
plt.show()

# 10. sns.pointplot
sns.pointplot(x="day", y="tip", data=df)
plt.show()

# 11. sns.regplot
sns.regplot(x="total_bill", y="tip", data=df)
plt.show()

# 12. sns.lmplot
sns.lmplot(x="total_bill", y="tip", data=df)

# 13. sns.heatmap
corr = df.corr(numeric_only=True)
sns.heatmap(corr, annot=True, cmap="coolwarm")
plt.show()

# 14. sns.clustermap
sns.clustermap(corr, annot=True)
plt.show()
```

```
# 15. sns.pairplot
sns.pairplot(df)
plt.show()

# 16. sns.jointplot
sns.jointplot(x="total_bill", y="tip", data=df, kind="hex")
plt.show()

# 17. sns.set_style
sns.set_style("whitegrid")
sns.boxplot(x="day", y="tip", data=df)
plt.show()

# 18. sns.set_context
sns.set_context("talk")
sns.scatterplot(x="total_bill", y="tip", data=df)
plt.show()

# 19. sns.color_palette
colors = sns.color_palette("pastel")
sns.barplot(x="sex", y="tip", data=df, palette=colors)
plt.show()

# 20. sns.despine
sns.boxplot(x="day", y="tip", data=df)
sns.despine()  # removes top and right border
plt.show()
```

Tensor flow

```
import tensorflow as tf
import numpy as np

print("=== Tensor Operations ===")

# 1. tf.constant
const_tensor = tf.constant([[1, 2], [3, 4]])
print("tf.constant:\n", const_tensor)

# 2. tf.Variable
var_tensor = tf.Variable([[5.0, 6.0], [7.0, 8.0]])
print("tf.Variable:\n", var_tensor)

# 3. tf.matmul
result = tf.matmul(const_tensor, var_tensor)
print("tf.matmul:\n", result)

# 4. tf.add
added = tf.add(const_tensor, 2)
print("tf.add:\n", added)

# 5. tf.reshape
reshaped = tf.reshape(const_tensor, (4, 1))
print("tf.reshape:\n", reshaped)

print("\n=== Model Building ===")

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D

# 6. tf.keras.Sequential
model = Sequential()

# 7. tf.keras.layers.Dense
model.add(Dense(units=64, activation='relu', input_shape=(100,)))

# 8. tf.keras.layers.Flatten
model.add(Flatten())

# 9. tf.keras.layers.Conv2D
model.add(Conv2D(32, (3, 3), activation='relu'))

# 10. tf.keras.layers.MaxPooling2D
model.add(MaxPooling2D(pool_size=(2, 2)))

print("Model Summary:")
model.summary()

print("\n=== Compiling and Training ===")

# Dummy data for training
X = np.random.random((100, 100))
y = np.random.randint(0, 2, 100)

# 11. model.compile
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

# 12. model.fit
model.fit(X, y, epochs=5, batch_size=10)

print("\n=== Evaluation and Prediction ===")

# 13. model.evaluate
loss, acc = model.evaluate(X, y)
print("Loss:", loss, "Accuracy:", acc)
```

```
14. model.predict
predictions = model.predict(X[:5])
print("Predictions (first 5):\n", predictions)

print("\n=== Data and Preprocessing ===")

# 15. tf.data.Dataset
dataset = tf.data.Dataset.from_tensor_slices((X, y))
print("tf.data.Dataset created:", dataset)

# 16. tf.keras.utils.to_categorical
y_cat = tf.keras.utils.to_categorical(y, num_classes=2)
print("One-hot encoded y:\n", y_cat[:5])

# 17. tf.image.resize
image = tf.random.normal([28, 28, 3])
resized_image = tf.image.resize(image, [64, 64])
print("Resized image shape:", resized_image.shape)

print("\n=== Callbacks ===")

# 18. tf.keras.callbacks.EarlyStopping
early_stop = tf.keras.callbacks.EarlyStopping(patience=3)

# 19. tf.keras.callbacks.ModelCheckpoint
checkpoint = tf.keras.callbacks.ModelCheckpoint('model_checkpoint.h5',
save_best_only=True)

print("Callbacks created: EarlyStopping & ModelCheckpoint")

print("\n=== Saving and Loading Model ===")

# 20. model.save and load_model
model.save('my_model.h5')
print("Model saved as 'my_model.h5'")

loaded_model = tf.keras.models.load_model('my_model.h5')
print("Model loaded from 'my_model.h5'")
```