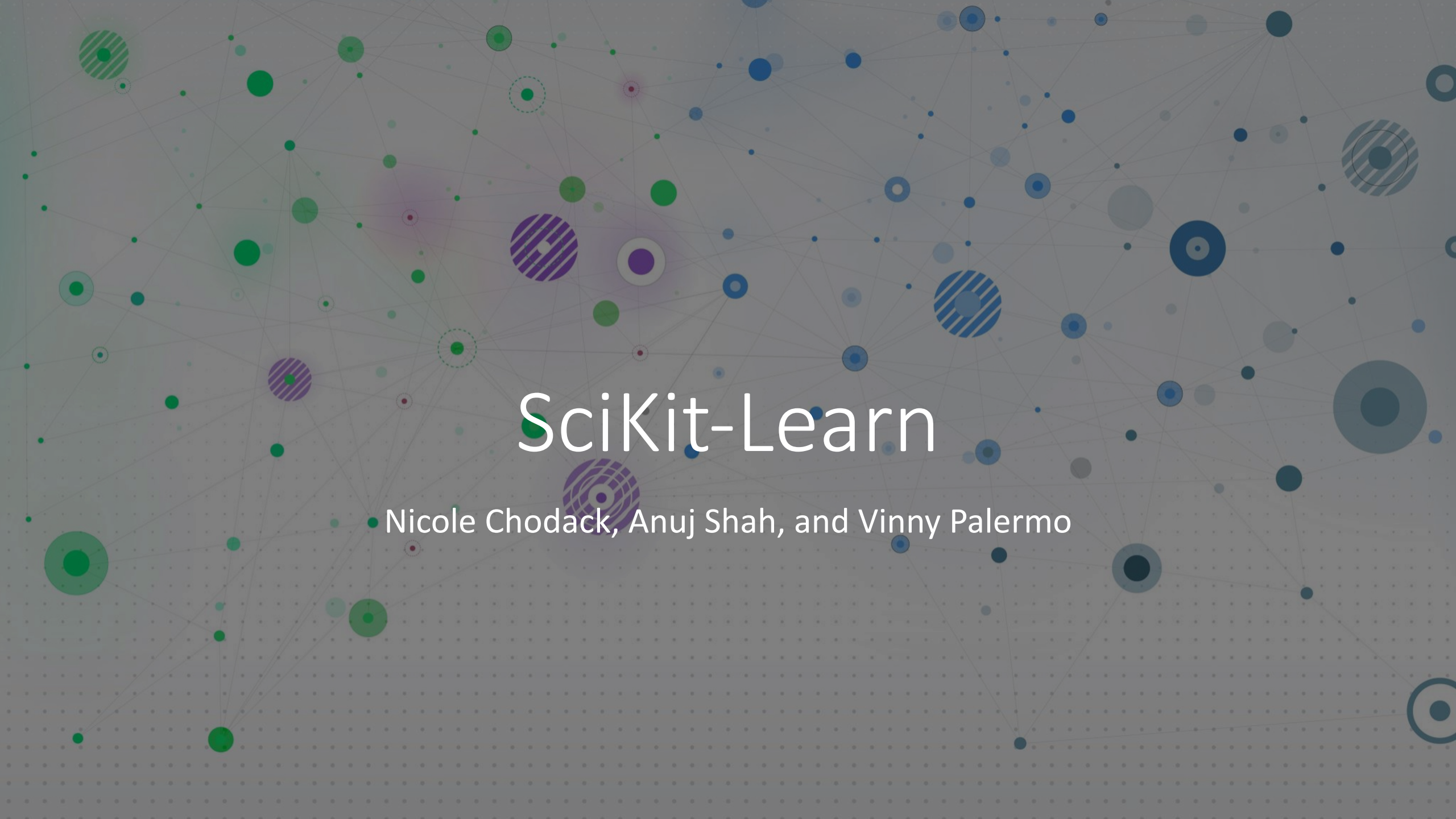
SciKit-Learn
Nicole Chodack, Anuj Shah, and Vinny Palermo

What is scikit-learn?

Popular Python library for data mining, data analysis, and machine learning.

Built on top of NumPy, SciPy, and Matplotlib

Comes with pre-built machine learning models and tools

Advantages and Drawbacks

- Advantages:
  - Easy to use
  - Well documented: plenty of resources and examples online
- Drawbacks:
  - Doesn’t support deep learning architectures. You would usually need to use TensorFlow, PyTorch, or Keras for deep learning.
  - Doesn’t leverage GPU for computations – can be a limitation for large datasets or computationally intensive tasks.

Installation

Installation

Tools and Applications

Preprocessing

- Preprocessing tools prepare data for modeling by transforming, scaling, or encoding features
  - Feature Scaling: scikit learn tools like ‘StandardScaler’ and ‘MinMaxScaler’ rescale numerical features to have specific properties(ex. zero mean and unit variance)
  - Feature Encoding: Transformers like ‘OneHotEncoder’ convert categorical variables into a numerical format suitable for machine learning models.
  - Handling Missing Data: Transformers like ‘Imputer’ help fill in missing values in datasets using specified strategies(ex. mean imputation)

Implementation - Preprocessing

```
from sklearn.preprocessing import StandardScaler

# Create an instance of the StandardScaler for feature scaling
scaler = StandardScaler()

# Fit and transform the training data
X_train_scaled = scaler.fit_transform(X_train)

# Transform the test data using the same scaler
X_test_scaled = scaler.transform(X_test)
```

Estimators

- These are the core components of scikit-learn, representing models for different machine learning tasks.
- This term encompasses classification, regression, clustering, and more.
- They have methods like ‘fit()’ for training on data and ‘predict()’ for making predictions.
  - The ‘fit()’ method takes the training data as input and computes the internal model parameters or structures based on the provided data.
  - The ‘predict()’ method allows you to apply the learned model to generate predictions.

Implementation - Estimators

```
from sklearn.linear_model import LogisticRegression

# Create an instance of the Logistic Regression estimator
model = LogisticRegression()

# Fit the model to training data
model.fit(X_train, y_train)

# Make predictions on new data
predictions = model.predict(X_test)
```

Model Selection and Evaluation

- Tools for splitting data in training and testing sets, performing cross-validation, and optimizing model hyperparameters.
  - Data Splitting: The ‘train_test_split’ function divides the dataset into training and testing sets, allowing you to assess model performance on unseen data.
  - Cross-Validation: The ‘cross_val_score’ function performs k-fold cross-validation to estimate a model’s performance across multiple subsets of data.
  - Hyperparameter Tuning: ‘GridSearchCV’ helps to search for the best combination of hyperparameters over a specified parameter grid for a given model.

Implementation – Model Selection

```
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Perform 5-fold cross-validation
scores = cross_val_score(model, X_train, y_train, cv=5)

# Perform hyperparameter tuning using GridSearchCV
param_grid = {'C': [0.1, 1, 10]}
grid_search = GridSearchCV(model, param_grid, cv=3)
grid_search.fit(X_train, y_train)
```

Classification and Regression Algorithms

- Specific type of estimator – scikit learn includes a variety of machine learning algorithms for classification and regression.
- Classification: Assigning items to categories or classes
  - Examples: RandomForestClassifier, KneighborsClassifier
- Regression: Predicting continuous values
  - Examples: LinearRegression, GradientBoostingRegressor
- Each algorithm has its own set of parameters and properties for its specific tasks, such as decision criteria for decision trees or kernel types for SVMs.

Implementation – Classification Algorithm

```
from sklearn.ensemble import RandomForestClassifier

# Create an instance of the RandomForestClassifier
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Fit the model to training data
model.fit(X_train, y_train)

# Make predictions on new data
predictions = model.predict(X_test)
```

Clustering Algorithms

- Algorithms for grouping similar data points into clusters without predefined categories.
  - Unsupervised Clustering: ‘KMeans’, ‘AgglomerativeClustering’

Implementation - Clustering

```
from sklearn.cluster import KMeans

# Create an instance of the KMeans clustering algorithm
kmeans = KMeans(n_clusters=3, random_state=42)

# Fit the model to the data
kmeans.fit(X)
```

Ensemble Methods

- Bagging: Creating multiple subsets of the original data, training a model on each subset, and averaging the predictions. Ex. ‘BaggingClassifier’, ‘BaggingRegressor’, ’RandomForestClassifier’
- Boosting: Training a series of models, where each subsequent model tries to correct the errors made by the previous ones. These models are then combined to make final prediction. Ex. ‘AdaBoost’, ‘GradientBoostingClassifier’, ‘XGBoost’
- Stacking: Several different models are trained and their predictions combined using another model to make the final prediction. Ex. ‘StackingClassifier’, ‘StackingRegressor’

Implementation - Boosting

AdaBoost works by adjusting the weights of misclassified instances and building a series of weak learners.

```python
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier

# Set up the AdaBoost classifier
# - Base estimator: A decision tree with depth = 1 (a "stump" - each tree has single decision node and two leaf nodes)
# - n_estimators: Number of weak learners to train (200 in this case)
# - algorithm: Use the SAMME.R real boosting algorithm ("Stagewise Additive Modeling using a Multiclass Exponential loss function")
# - learning_rate: Weight of each stump. A value of 0.5 means each subsequent model corrects half of the mistakes of its predecessor.
ada = AdaBoostClassifier(
    DecisionTreeClassifier(max_depth=1),
    n_estimators=200,
    algorithm="SAMME.R",
    learning_rate=0.5
)
ada.fit(X_train, y_train)
```

[Kaggle-XGBoost example]

Dimensionality Reduction

- Algorithms for reducing the dimensionality of datasets while preserving important information.
  - Feature Transformation: Techniques like ‘PCA’ transform high-dimensional data into a lower-dimensional representation, making it easier to visualize and analyze.

Implementation - PCA

```
from sklearn.decomposition import PCA

# Create an instance of PCA with 2 components
pca = PCA(n_components=2)

# Fit and transform the data to reduce dimensions
X_reduced = pca.fit_transform(X)
```

Metrics

- Help evaluate the performance of machine learning models.
  - Classification Metrics: Metrics like accuracy, precision, recall, F1-score assess classification model performance.
  - Regression Metrics: Metrics like mean squared error (MSE), mean absolute error(MAE), and R-squared measure regression model performance.
  - Clustering Metrics: Metrics like silhouette score can be used to evaluate clustering results.

Implementation - Metrics

```
from sklearn.metrics import accuracy_score, precision_score

# Calculate accuracy
accuracy = accuracy_score(true_labels, predicted_labels)

# Calculate precision
precision = precision_score(true_labels, predicted_labels)
```

Datasets

- A collection of built-in datasets for practice and experimentation
  - Ex. Iris dataset commonly used for machine learning practice

Pipeline

- Pipeline class in scikit-learn is a tool for streamlining a lot of the routine processes. It bundles together a sequence of data processing steps and modeling into a single object.
- Each step in the pipeline is represented by a tuple: the first element of the tuple is a name (a string), and the second element is an instance of the transformer or model.
- Ex. Pipeline example

Demo

Resources

- scikit-learn documentation
- https://www.tutorialspoint.com/scikit_learn/scikit_learn_introduction.htm
- https://www.geeksforgeeks.org/learning-model-building-scikit-learn-python-machine-learning-library/