# Сравнение библиотек для машинного обучения: scikit-learn, TensorFlow и PyTorch  
  
В этом проекте мы будем сравнивать три популярных библиотеки для машинного обучения: scikit-learn, TensorFlow и PyTorch. Мы реализуем задачи классификации и регрессии с использованием каждой из этих библиотек и проведем анализ их производительности и удобства использования.  
  
## Содержание  
  
- Введение  
- Установка  
- Структура проекта  
- Реализация  
- Сравнение библиотек  
- Заключение  
  
  
## Введение  
  
В последние годы машинное обучение стало важным инструментом в различных областях. Существует множество библиотек, каждая из которых имеет свои преимущества и недостатки. В данном проекте мы сосредоточимся на следующих библиотеках:  
  
- scikit-learn: Простая и эффективная библиотека для анализа данных, работающая на высоком уровне.  
- TensorFlow: Мощная библиотека для выполнения операций над тензорами, хорошо подходящая для глубокого обучения.  
- PyTorch: Интуитивно понятная библиотека, которая предлагает динамические вычислительные графы, что делает её популярной среди исследователей.  
  
## Установка  
  
Для установки необходимых библиотек, выполните следующие команды:  
  
```bash  
pip install scikit-learn  
pip install tensorflow  
pip install torch torchvision  
```  
  
## Структура проекта  
  
  
/project-root  
│  
├── /data # Данные для классификации и регрессии  
│  
├── /notebooks # Jupyter notebook с реализациями  
│  
├── /src # Исходный код проекта  
│ ├── classifiers.py # Реализация классификаторов  
│ └── regressors.py # Реализация регрессоров  
│  
└── README.md # Этот файл  
```  
  
## Реализация  
  
### Классификация  
  
В файле classifiers.py реализованы следующие методы классификации для каждой из библиотек:  
  
python  
def classify\_with\_sklearn(X\_train, y\_train, X\_test):  
  
Выполняет классификацию с использованием scikit-learn.  
  
Parameters:  
X\_train: Обучающие данные  
y\_train: Метки обучающих данных  
X\_test: Тестовые данные  
  
Returns:  
The predicted labels for the test data.  
  
# Импортируйте необходимый классификатор из scikit-learn  
from sklearn.ensemble import RandomForestClassifier  
  
# Инициализация и обучение модели  
model = RandomForestClassifier()  
model.fit(X\_train, y\_train)  
  
# Прогнозирование  
return model.predict(X\_test)  
```  
  
### Регрессия  
  
Аналогично, в файле regressors.py реализуем задачи регрессии с использованием всех трех библиотек:  
  
python  
def regress\_with\_pytorch(X\_train, y\_train, X\_test):  
  
Выполняет регрессию с использованием PyTorch.  
  
Parameters:  
X\_train: Обучающие данные  
y\_train: Целевые значения  
X\_test: Тестовые данные  
  
Returns:  
The predicted values for the test data.  
  
import torch  
import torch.nn as nn  
  
# Определяем модель  
class LinearRegressionModel(nn.Module):  
def \_\_init\_\_(self):  
super(LinearRegressionModel, self).\_\_init\_\_()  
self.linear = nn.Linear(X\_train.shape[1], 1)  
  
def forward(self, x):  
return self.linear(x)  
  
# Преобразуем данные в тензоры  
X\_train\_tensor = torch.FloatTensor(X\_train)  
y\_train\_tensor = torch.FloatTensor(y\_train).view(-1, 1)  
  
# Инициализация модели и оптимизатора  
model = LinearRegressionModel()  
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)  
  
# Обучение модели (пример 100 эпох)  
for epoch in range(100):  
model.train()  
optimizer.zero\_grad()  
predictions = model(X\_train\_tensor)  
loss = nn.MSELoss()(predictions, y\_train\_tensor)  
loss.backward()  
optimizer.step()  
  
# Прогнозирование  
X\_test\_tensor = torch.FloatTensor(X\_test)  
return model(X\_test\_tensor).detach().numpy()  
  
  
## Сравнение библиотек  
  
После реализации всех методов, мы сравним производительность каждой из библиотек по следующим критериям:  
  
1. Время обучения  
2. Точность прогнозов  
3. Удобство использования и читаемость кода  
  
## Заключение

В данном проекте мы внедрили несколько методов классификации и регрессии, используя scikit-learn, TensorFlow и PyTorch, и провели их сравнение. Данный анализ помогает выбрать правильный инструмент для конкретной задачи машинного обучения.