

A Basic AI Model Implementation

A step-by-step walk-through of building a basic machine learning model in Python, using a random forest classifier on customer churn data.

Siddhant Shah 25 Jun 2024 3 min

Machine learning

The example used here analyses Customer Churn Data using a Random Forrest Classifier. This is used for demonstrative purposes only and the steps, libraries and data used by you may vary.

The Implementation Process

Data Collection & Processing

- Start by collecting data from some verified data source.
- Then pre-process the entire dataset (if needed) to ensure uniformity.

Example Code

```
import pandas as pd
from sklearn.preprocessing import OrdinalEncoder
oe = OrdinalEncoder()

data = pd.read_csv('Data/Customer Churn.csv')

data['Gender'] = oe.fit_transform(data[['Gender']])
data['EmailOptIn'] = oe.fit_transform(data[['EmailOptIn']])
data['PromotionResponse'] = oe.fit_transform(data[['PromotionResponse']])

X = data.iloc[:, :-1]
y = data.iloc[:, -1]
```

- Here, we first read the data that is stored in a CSV (Comma-separated Values) file.
- The data is then pre-processed into a format that can be understood and used by the model. The processing may vary depending on the model you use.
- Finally, we separate the data into the measured values and the goal variable.

- Separate data into training and testing sets. Testing sets will only be used after the model has been selected. Data leakage is unacceptable.

Example Code

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.2,
    stratify=X['Gender'],
    random_state=1
)
```

- `test_size=0.2` denotes that we are using 20% of the data for the testing set.

Implement Models and Compare Performance

- Choose some [Machine Learning Models](#). These will be implemented in Python using the Scikit-learn library, resources for which can be found at the end.
- Implement your selected models with various hyper-parameter values for each model.

Example Code

- Here we choose to implement a *Random Forrest Classifier*.

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()

grid = {
    'n_estimators': [25 * i for i in range(1, 10 + 1)],
    'max_features': ['sqrt', 'log2'],
    'max_depth': [3 * i for i in range(1, 5 + 1)],
    'max_leaf_nodes': [2, 5, 10, 20, 50],
}
```

- The dictionary `grid` contains the names of the hyper-parameters that can be tuned for this particular model.

- Train each such configuration on the training data set and evaluate it. Evaluation will consist of the following steps
 - After training, use the data from a sub-dataset to make predictions.
 - Use these predictions and compare them with the actual values from the data. (This process is called validation.)
 - Based on the predictions and the actual values, compute values of different metrics such as accuracy, recall, mean-squared error, or others depending on the use case.

Example Code

```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RepeatedStratifiedKFold

cv = RepeatedStratifiedKFold(
    n_splits=10,
    n_repeats=3,
    random_state=1
)
grid_search = GridSearchCV(
    estimator=model,
    param_grid=grid,
    n_jobs=-1,
    cv=cv,
    scoring='accuracy'
)

grid_result = grid_search.fit(X_train, y_train)
print(f"Highest Accuracy: {grid_result.best_score_:.4f}")
print(f"Hyper Parameter Values: {grid_result.best_params_}")
```

- Here we use functions called `RepeatedStratifiedKFold` and `GridSearchCV` as `RepeatedStratifiedKFold` creates a validation set from the training data on its own, without any data leakage and `GridSearchCV` automatically varies across all the hyper-parameter values, trains the given model and evaluates the given score over the validation dataset.
- At the end, we use the last two `print` statements to get the highest accuracy achieved across all the hyper-parameter values and to get the respective values as well.

Insights & Analysis

- Based on the previous metric, choose the ideal model configuration.

Example Code

```
rf = RandomForestClassifier(  
    n_estimators=200,  
    max_features='log2',  
    max_leaf_nodes=2,  
    max_depth=12,  
    random_state=1  
)  
  
rf_model = rf.fit(X_train, y_train)
```

- For the selected model, explain what the values of the metrics means and what the corresponding values imply. Explain why the particular metrics were chosen.
 - Results can shown in terms of Confusion Matrices, variations in metrics with change in values, etc.
 - Possible insights can include improved performance in a specific use-case, a hypothesis regarding the structure of the data based on the model, etc.

Possible topics

1. Pneumonia or Diabetes Prediction:

Here, measured values will be vitals such as heart rate, body temperature, previous medical history, etc. The goal variable will be a binary value (0 or 1) denoting the absence or presence of the disease.

2. Wildfire Threat:

Here the measured values will include distance from nearby forest fire, settlements, water bodies, etc. The goal variable will be a score denoting how likely a wildfire is.

The values mentioned for each topic are not exhaustive. Other values can and should be used. The student should be able to justify the use of every value through some direct or indirect logic.

Resources

1. Intro to Python (Official Python Documentation): <https://www.python.org>
2. Scikit-Learn Documentation: <https://scikit-learn.org/stable/>

DISCLOSURE

The views expressed here are my own personal opinions. This is **not investment advice** and should not be relied upon as such. Nothing here is connected to, endorsed by, or written on behalf of Rosenblatt Securities.

© 2026 Siddhant Shah