__/__/____

Predict Employee turnover with scikit-learn.

→ It is a free software machine learning library for Python programming language.

→ Predict if employees are likely to quit or not.

→ Quit → 1 means yes.

TASK-1 IMPORT LIBRARY

TASK-2 EXPLORATORY DATA ANALYSIS

```
hr = pd.read_csv('data/employee_data.csv') → Import
hr.head() → First few entries
```

```
hr.profile_report(title="Data") → Overview report of data
```

x-axis & y-axis, bar graph

```
pd.crosstab(hr.Salary, hr.quit).plot(kind='bar')
plt.title("Turnover freq on Salary Bracket")
plt.xlabel('Salary')
plt.ylable('Frequency of turnover')
plt.show()
```

TASK-3 Encode Categorical feature → The variables are categorical i.e. string as object but scikit-learn and many others can't process these so encoding is done.

We will use one ~~hot~~ not encoding. Matlab jaise ek employee ke dept mein accounting hai toh table banega and uss accounting ke age 1 baki ke age 0.

DANGER

pd.get_dummies() → age direct use krenge toh data loose hoga.

```
category_variables = ['department', 'salary']   # name save rakhiye taki data na loose ho
for var in category_variables:
    category_list = pd.get_dummies(hr[var], prefix=var)
    hr = hr.join(category_list)
```

```
hr.head()
```

Ab hum main salary and department delete krdenge.

```
hr.drop(columns = ['department', 'salary'],
        axis = 1, inplace = True)
```

↓ isko matlab vertical

PTO →

__/__/____

TASK 4 - Visualize Class Imbalance. → Agar sab kuch ke bawjood testing mein fail ho rahe ho toh ye problem ho sakti hai. Ek baar check kar lena chahiye.

```
from yellowbrick.target import ClassBalance
plt.style.use('ggplot')
plt.rcParams['figure.figsize'] = (12,8)
```

value 0 value 1

```
visualizer = ClassBalance(labels=['stayed', 'quit'])
visualizer.fit(hr.quit)
visualizer.poof()
```

Kisko 0 or 1

Apply → Solution

Stratified sampling when creating training & validation sets

Apne example mein bar graph mein bohut difference hai. So, we have a class imbalance problem. More employees who stayed than quit

Task 5 - Create Training & Test Sets

Separate data into feature matrix (called X) and target vector → sirf quit ki hai target.

locate not quit

```
X = hr.loc[:, hr.columns != 'quit']
y = hr.quit
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split
    (X, y, random_state=0, test_size=0.2, stratify=y)
```

Scanned with CamScanner

stratify y ensures class distribution is approx that of y. i.e
Eg → Y contains 25% - '0' and 75% - '1' then y-train & y-test will have 25% - '0' and 75% '1'

TASK 6 → Build Decision Tree Classifier with Interactive console

To make interactive we use @interact

```
@interact
def plot_tree(crit=['gini', 'entropy'],          — Dropdown
              split=['best', 'random']
              depth=IntSlider(min=1, max=30, value=2
              continuous_update=False)):
                                          ↑ default
```

Aur bhi hai check code

continuous_update ↓

less computation and overhead → False → jab drag karke release karoge tab update hoga

True → jaise drag karoge vaise update hoga

scikit learn decision tree classifier → estimator = DecisionTreeClassifier(random_state=0,
criterion = crit,
splitter = split,
max_depth = depth,
min_samples_split = min_split,
min_samples_leaf = min_leaf)

```
estimator.fit(x_train, y_train)  ← training data pe
print('Decision Tree training Accuracy = {:.3f}'.
      format(accuracy_score(y_train,
      estimator.predict(x_train))
```

:.3f mattlab upto 3 decimal places
accuracy_score → sklearn mein hai
y_train → because training accuracy
estimator.predict(x_train) → ~~kyu~~ kyunki fit kar liya hai data.

package name — helper functn — predicted model

graph = Source(tree.export_graphviz(estimator,

no output file → out_file = None,

display values in tree → feature_names = X_train.columns,
class_names = ['stayed', 'quit'],
filled = True))

display(Image(data = graph.pipe(format = 'png')))

TASK 8 → Build Interactive Random Forest Classifier

See Code Comments.

TASK-9 Feature Estimator

and
Task 6 & 8 main estimator return krwao

ef = Copy returned estimator from Random forest (Task-8)

Visualize viz = FeatureImportances(ef)
viz.fit(X_train, y_train)
viz.show();