[1]:
```python
# ========================================
# 3) Model Building – Upwork Jobs
# ========================================
# Bu bölümde CSV verisinden makine öğrenmesi modelleri oluşturuluyor.
# Amaç: Upwork ilanlarının türünü ve ücretini tahmin edebilmek.

import os, numpy as np, pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import (
    accuracy_score, f1_score, classification_report,
    r2_score, mean_absolute_error, root_mean_squared_error
)
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
```

[2]:
```python
# ---------- 1) Tek CSV Dosyasını Yükleme & Basit Ön İşleme ----------
DATA_DIR = r"C:\Users\asile\Desktop\Upwork_Project22222\Upwork_Project\data"
CSV_FILE = "all_upwork_jobs_2024-02-07-2024-03-24.csv"  # Dosya adını değiştirdiysen burada güncelle
csv_path = os.path.join(DATA_DIR, CSV_FILE)

# Dosya mevcut mu kontrol et
if not os.path.isfile(csv_path):
    raise FileNotFoundError(f"CSV not found at: {csv_path}")
```

[3]:
```python
# CSV dosyasını oku
df = pd.read_csv(csv_path, low_memory=False)
print("Loaded shape:", df.shape)
print("Columns:", list(df.columns))
```

```
Loaded shape: (244828, 8)
Columns: ['title', 'link', 'published_date', 'is_hourly', 'hourly_low', 'hourly_high', 'budget', 'country']
```

[7]:
```
# Metin ve kategorik sütunları standart hale getir
df["title"] = df.get("title","").astype(str).fillna("")
df["country"] = df.get("country","unknown").astype(str).fillna("unknown")
```

[8]:
```
# Büyük veri setlerinde işlem hızını artırmak için örnekleme (örneğin 245k satır → 50k satır)
SAMPLE_MAX = 50000  # Bilgisayarın yavaşsa 30000 yapabilirsin
df_sample = df.sample(SAMPLE_MAX, random_state=42) if len(df) > SAMPLE_MAX else df.copy()
print("Working sample shape:", df_sample.shape)
```

```
Working sample shape: (50000, 9)
```

```python
        pipe = Pipeline([("prep", pre), ("model", model)])  # Ön işleme + model zinciri
        pipe.fit(X_train, y_train)  # Modeli eğit
        y_pred = pipe.predict(X_test)  # Tahmin yap
        acc = accuracy_score(y_test, y_pred)  # Doğruluk oranı
        f1 = f1_score(y_test, y_pred)         # F1 skoru
        cls_rows.append({"Model": name, "ACC": acc, "F1": f1})
        print(f"[CLS] {name}: ACC={acc:.4f}  F1={f1:.4f}")
        print(classification_report(y_test, y_pred, digits=3))
    # Sonuç tablosu
    cls_results = pd.DataFrame(cls_rows).sort_values("F1", ascending=False).reset_index(drop=True)
    print("\n[CLS] Summary:\n", cls_results)
else:
    print("\n[Classification] Skipped: 'is_hourly' not available or not binary.")
```

```
[Classification] Train: (40000, 2) Test: (10000, 2)
[CLS] LogReg: ACC=0.6548  F1=0.7208
              precision    recall  f1-score   support

           0      0.614     0.495     0.548      4228
           1      0.676     0.772     0.721      5772

    accuracy                          0.655     10000
   macro avg      0.645     0.633     0.634     10000
weighted avg      0.650     0.655     0.648     10000

[CLS] RFCls: ACC=0.6535  F1=0.7195
              precision    recall  f1-score   support

           0      0.612     0.494     0.547      4228
           1      0.675     0.770     0.720      5772

    accuracy                          0.653     10000
   macro avg      0.643     0.632     0.633     10000
weighted avg      0.648     0.653     0.646     10000


[CLS] Summary:
     Model     ACC        F1
0  LogReg  0.6548  0.720802
1   RFCls  0.6535  0.719547
```

```python
        models_reg = {
            "LinReg": LinearRegression(),
            "RFReg": RandomForestRegressor(n_estimators=300, random_state=42, n_jobs=-1),
        }
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
        print(f"\n[Regression-Fixed] Rows={len(dreg)}  Train:{X_train.shape}  Test:{X_test.shape}")
        reg_rows = []
        for name, model in models_reg.items():
            pipe = Pipeline([("prep", pre), ("model", model)])
            pipe.fit(X_train, y_train)
            y_pred = pipe.predict(X_test)
            # Performans metrikleri:
            r2  = r2_score(y_test, y_pred)  # Modelin açıklama gücü (0-1 arası)
            mae = mean_absolute_error(y_test, y_pred)  # Ortalama hata miktarı
            rmse= root_mean_squared_error(y_test, y_pred)  # Kök ortalama kare hata
            reg_rows.append({"Model": name, "R2": r2, "MAE": mae, "RMSE": rmse})
            print(f"[REG-Fixed] {name}: R2={r2:.4f}  MAE={mae:.2f}  RMSE={rmse:.2f}")
        # Sonuçları tablo halinde yazdır
        regA_results = pd.DataFrame(reg_rows).sort_values("RMSE").reset_index(drop=True)
        print("\n[REG-Fixed] Summary:\n", regA_results)
        regA_done = True

if not regA_done:
    print("\n[Regression-Fixed] Skipped: need 'budget' with is_hourly==0 rows.")
```

```
[Regression-Fixed] Rows=21104  Train:(16883, 2)  Test:(4221, 2)
[REG-Fixed] LinReg: R2=-0.1525  MAE=956.41  RMSE=2532.28
[REG-Fixed] RFReg: R2=-0.0548  MAE=601.16  RMSE=2422.63

[REG-Fixed] Summary:
     Model        R2         MAE         RMSE
0   RFReg -0.054828  601.157768  2422.625345
1  LinReg -0.152476  956.411072  2532.278216
```

```
        print(f"\n[Regression-Hourly] Rows={len(dreg_h)}  Train:{X_train.shape}  Test:{X_test.shape}")
        reg_rows = []
        for name, model in models_reg.items():
            pipe = Pipeline([("prep", pre), ("model", model)])
            pipe.fit(X_train, y_train)
            y_pred = pipe.predict(X_test)
            r2  = r2_score(y_test, y_pred)
            mae = mean_absolute_error(y_test, y_pred)
            rmse= root_mean_squared_error(y_test, y_pred)
            reg_rows.append({"Model": name, "R2": r2, "MAE": mae, "RMSE": rmse})
            print(f"[REG-Hourly] {name}: R2={r2:.4f}  MAE={mae:.2f}  RMSE={rmse:.2f}")
        regB_results = pd.DataFrame(reg_rows).sort_values("RMSE").reset_index(drop=True)
        print("\n[REG-Hourly] Summary:\n", regB_results)
        regB_done = True

if not regB_done:
    print("\n[Regression-Hourly] Skipped: need 'hourly_rate' with is_hourly==1 rows.")
```

```
[Regression-Hourly] Rows=20253  Train:(16202, 2)  Test:(4051, 2)
[REG-Hourly] LinReg: R2=0.0053  MAE=17.90  RMSE=28.55
[REG-Hourly] RFReg: R2=0.1001  MAE=14.95  RMSE=27.16

[REG-Hourly] Summary:
     Model        R2        MAE       RMSE
0   RFReg  0.100076  14.951666  27.160147
1  LinReg  0.005289  17.900175  28.554699
```

[13]:
```
print("\nDone. işlem tamamlandı")  # Tüm işlemler tamamlandı
```

```
Done. işlem tamamlandı
```