NYC Taxi Trip Duration — Mini POC (Adaptable to Indore)

This PDF includes full documentation AND AI model code (Notebook + Streamlit).

Jupyter Notebook Code (01_nyc_trip_duration_poc.ipynb)

```
# cell 1: imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_log_error
import lightgbm as lgb
import joblib
from geopy.distance import great_circle
import folium

# cell 2: load data
df = pd.read_csv('../data/raw/nyc_train.csv', nrows=200000)
df.shape

# cell 3: cleaning + features
df = df[df['trip_duration']>0].copy()
df['pickup_datetime'] = pd.to_datetime(df['pickup_datetime'])
df['hour'] = df['pickup_datetime'].dt.hour
df['dayofweek'] = df['pickup_datetime'].dt.dayofweek

def haversine_row(row):
    return great_circle((row['pickup_latitude'], row['pickup_longitude']),
                        (row['dropoff_latitude'], row['dropoff_longitude'])).km
df['dist_km'] = df.apply(haversine_row, axis=1)
df = df[(df['dist_km'] < 100) & (df['trip_duration'] < 24*3600)]

# train/test
features = ['vendor_id','passenger_count','dist_km','hour','dayofweek']
X = df[features]
y = df['trip_duration']
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

# model
train_data = lgb.Dataset(X_train, label=y_train)
valid_data = lgb.Dataset(X_val, label=y_val)
params = {'objective':'regression','metric':'rmse','learning_rate':0.1,'num_leaves':31}
model = lgb.train(params, train_data, valid_sets=[train_data, valid_data],
                  early_stopping_rounds=50, num_boost_round=500)

# evaluate
preds = model.predict(X_val)
rmsle = np.sqrt(mean_squared_log_error(y_val.clip(1), preds.clip(1)))
print("RMSLE:", rmsle)

# save model
joblib.dump(model, '../models/lgbm_model.pkl')
```

Streamlit App Code (streamlit_app.py)

```
import streamlit as st
import pandas as pd
import numpy as np
import joblib
from geopy.distance import great_circle
import streamlit.components.v1 as components
import folium

st.set_page_config(page_title="Taxi ETA POC", layout="wide")
st.title("Taxi Trip Duration POC (NYC) — Adaptable to Indore")

@st.cache_resource
def load_model():
    return joblib.load('../models/lgbm_model.pkl')
model = load_model()

st.sidebar.header("Trip inputs")
vendor_id = st.sidebar.selectbox("vendor_id", [1,2])
passenger_count = st.sidebar.slider("passenger_count", 0, 6, 1)
pickup_lat = st.sidebar.number_input("pickup_latitude", value=40.7612, format="%.6f")
pickup_lon = st.sidebar.number_input("pickup_longitude", value=-73.9822, format="%.6f")
drop_lat = st.sidebar.number_input("dropoff_latitude", value=40.7128, format="%.6f")
drop_lon = st.sidebar.number_input("dropoff_longitude", value=-74.0060, format="%.6f")
hour = st.sidebar.slider("hour_of_day (0-23)", 0, 23, 12)
dayofweek = st.sidebar.slider("day_of_week (0=Mon)", 0, 6, 2)

def compute_dist_km(pick, drop):
    return great_circle(pick, drop).km

if st.sidebar.button("Predict ETA"):
    dist = compute_dist_km((pickup_lat, pickup_lon),(drop_lat,drop_lon))
    X = pd.DataFrame([{
        'vendor_id': vendor_id,
        'passenger_count': passenger_count,
        'dist_km': dist,
        'hour': hour,
        'dayofweek': dayofweek
    }])
    pred = model.predict(X)[0]
    st.metric("Predicted trip duration (seconds)", f"{int(pred):,} s")

    m = folium.Map(location=[(pickup_lat+drop_lat)/2,(pickup_lon+drop_lon)/2], zoom_start=12)
    folium.Marker([pickup_lat,pickup_lon], popup="Pickup", icon=folium.Icon(color='green')).add_to(m
    folium.Marker([drop_lat,drop_lon], popup="Dropoff", icon=folium.Icon(color='red')).add_to(m)
    folium.PolyLine([[pickup_lat,pickup_lon],[drop_lat,drop_lon]], weight=2).add_to(m)
    m.save('tmp_map.html')
    HtmlFile = open('tmp_map.html','r',encoding='utf-8')
    source_code = HtmlFile.read()
    components.html(source_code, height=400)
```