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.linear\_model import LinearRegression  
from sklearn.ensemble import RandomForestRegressor  
from sklearn.metrics import r2\_score, mean\_squared\_error  
from sklearn.impute import SimpleImputer  
from scipy import stats  
  
# Load the dataset  
df = pd.read\_csv('uber.csv')  
# Display the first few rows of the dataset  
print(df.head())  
  
# Check for missing values  
print(df.isnull().sum())  
# Convert pickup\_datetime to datetime objects  
df['pickup\_datetime'] = pd.to\_datetime(df['pickup\_datetime'])  
  
# Impute missing values for numeric columns (example using mean imputation)  
numeric\_columns = df.select\_dtypes(include=[np.number]).columns  
imputer = SimpleImputer(strategy='mean')  
df[numeric\_columns] = imputer.fit\_transform(df[numeric\_columns])  
  
# Drop rows with missing target values  
df.dropna(subset=['fare\_amount'], inplace=True)  
  
# Extract useful features from datetime  
df['pickup\_year'] = df['pickup\_datetime'].dt.year  
df['pickup\_month'] = df['pickup\_datetime'].dt.month  
df['pickup\_day'] = df['pickup\_datetime'].dt.day  
df['pickup\_hour'] = df['pickup\_datetime'].dt.hour  
  
# Drop unnecessary columns  
df.drop(columns=['pickup\_datetime', 'key'], inplace=True)  
# Calculate the correlation matrix  
corr\_matrix = df.corr()  
  
# Plot the heatmap  
plt.figure(figsize=(10, 8))  
sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm')  
plt.show()  
  
# Split the dataset into features and target variable  
X = df.drop(columns=['fare\_amount'])  
y = df['fare\_amount']  
  
# Split the data into training and testing sets  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Linear Regression  
lr\_model = LinearRegression()  
lr\_model.fit(X\_train, y\_train)  
y\_pred\_lr = lr\_model.predict(X\_test)  
  
# Random Forest Regression  
rf\_model = RandomForestRegressor(n\_estimators=100, random\_state=42)  
rf\_model.fit(X\_train, y\_train)  
y\_pred\_rf = rf\_model.predict(X\_test)  
  
  
# Evaluate Linear Regression  
r2\_lr = r2\_score(y\_test, y\_pred\_lr)  
rmse\_lr = np.sqrt(mean\_squared\_error(y\_test, y\_pred\_lr))  
  
# Evaluate Random Forest Regression  
r2\_rf = r2\_score(y\_test, y\_pred\_rf)  
rmse\_rf = np.sqrt(mean\_squared\_error(y\_test, y\_pred\_rf))  
  
# Print the evaluation metrics  
print(f"Linear Regression - R²: {r2\_lr}, RMSE: {rmse\_lr}")  
print(f"Random Forest Regression - R²: {r2\_rf}, RMSE: {rmse\_rf}")