```
import pandas as pd
import numpy as np
import matplotlib as plt
import seaborn as sns

df = pd.read_csv("uber.csv")


df.head()


df.info()


df.describe()



df.shape


df.isnull().sum()


df.dropna(inplace = True)



df.isnull().sum()



df.drop(labels='Unnamed: 0',axis=1,inplace=True)
df.drop(labels='key',axis=1,inplace=True)



df.head()


df["pickup_datetime"] = pd.to_datetime(df["pickup_datetime"])



df.describe()


import warnings
warnings.filterwarnings("ignore")
sns.distplot(df['fare_amount'])



sns.distplot(df['pickup_latitude'])


# In[17]:
```

```
sns.distplot(df['pickup_longitude'])


# In[18]:


sns.distplot(df['dropoff_longitude'])


# In[19]:


sns.distplot(df['dropoff_latitude'])


# In[20]:


#creating a function to identify outliers

def find_outliers_IQR(df):
   q1 = df.quantile(0.25)
   q3 = df.quantile(0.75)
   IQR = q3-q1
   outliers = df[((df<(q1-1.5*IQR)) | (df>(q3+1.5*IQR)))]
   return outliers


# In[21]:


#getting outlier details for column "fair_amount" using the above
function

outliers = find_outliers_IQR(df["fare_amount"])
print("number of outliers: "+ str(len(outliers)))
print("max outlier value: "+ str(outliers.max()))
print("min outlier value: "+ str(outliers.min()))
outliers


# In[22]:


#you can also pass two columns as argument to the function (here
"passenger_count" and "fair_amount")

outliers = find_outliers_IQR(df[["passenger_count","fare_amount"]])
outliers


# In[23]:


#upper and lower limit which can be used for capping of outliers

upper_limit = df['fare_amount'].mean() + 3*df['fare_amount'].std()
```

```
print(upper_limit)
lower_limit = df['fare_amount'].mean() - 3*df['fare_amount'].std()
print(lower_limit)


# # 3.Check the correlation

# In[26]:


#creating a correlation matrix

corrMatrix = df.corr()
sns.heatmap(corrMatrix, annot=True)
plt.show()


# In[27]:


#splitting column "pickup_datetime" into 5 columns: "day", "hour",
"month", "year", "weekday"
#for a simplified view

import calendar
df['day']=df['pickup_datetime'].apply(lambda x:x.day)
df['hour']=df['pickup_datetime'].apply(lambda x:x.hour)
df['month']=df['pickup_datetime'].apply(lambda x:x.month)
df['year']=df['pickup_datetime'].apply(lambda x:x.year)
df['weekday']=df['pickup_datetime'].apply(lambda x:
calendar.day_name[x.weekday()])
df.drop(['pickup_datetime'],axis=1,inplace=True)


# In[28]:


#label encoding (categorical to numerical)

df.weekday =
df.weekday.map({'Sunday':0,'Monday':1,'Tuesday':2,'Wednesday':3,'Thursday
':4,'Friday':5,'Saturday':6})


# In[29]:


df.head()


# In[30]:


df.info()


# In[31]:
```

```
#splitting the data into train and test

from sklearn.model_selection import train_test_split


# In[32]:


#independent variables (x)

x=df.drop("fare_amount", axis=1)
x


# In[33]:


#dependent variable (y)

y=df["fare_amount"]


# In[34]:


x_train,x_test,y_train,y_test =
train_test_split(x,y,test_size=0.2,random_state=101)


# In[35]:


x_train.head()


# In[36]:


x_test.head()


# In[37]:


y_train.head()


# In[38]:


y_test.head()


# In[39]:


print(x_train.shape)
```

```
print(x_test.shape)
print(y_test.shape)
print(y_train.shape)


# # 4.Implementing linear regression and random forest regression models

# In[40]:


#Linear Regression

from sklearn.linear_model import LinearRegression
lrmodel=LinearRegression()
lrmodel.fit(x_train, y_train)


# In[41]:


predictedvalues = lrmodel.predict(x_test)


# In[42]:


#Random Forest Regression

from sklearn.ensemble import RandomForestRegressor
rfrmodel = RandomForestRegressor(n_estimators=100, random_state=101)


# In[43]:


rfrmodel.fit(x_train,y_train)
rfrmodel_pred= rfrmodel.predict(x_test)


# # 5. Evaluate the models and compare their respective scores like R2,
RMSE, etc.

# In[44]:


#Calculating the value of RMSE for Linear Regression

from sklearn.metrics import mean_squared_error
lrmodelrmse = np.sqrt(mean_squared_error(predictedvalues, y_test))
print("RMSE value for Linear regression is", lrmodelrmse)


# In[45]:


#Calculating the value of RMSE for Random Forest Regression

rfrmodel_rmse=np.sqrt(mean_squared_error(rfrmodel_pred, y_test))
```

```
print("RMSE value for Random forest regression is ",rfrmodel_rmse)

# In[ ]:
```