Predict the price of the Uber ride from a given pickup point to the agreed drop-off location.

Perform following tasks:

1. Pre-process the dataset.
2. Identify outliers.
3. Check the correlation.
4. Implement linear regression and random forest regression models.
5. Evaluate the models and compare their respective scores like R2, RMSE, etc.

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```
df = pd.read_csv("uber.csv")
```

In [ ]:
```
df.head()
```

In [ ]:
```
df.info()
```

In [ ]:
```
df.shape
```

In [ ]:
```
df.isnull().sum()
```

In [ ]:
```
df.dropna(inplace = True)
```

In [ ]:
```
df.isnull().sum()
```

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

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

In [ ]:
```
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)
```

In [ ]:
```
df["pickup_datetime"] = pd.to_datetime(df["pickup_datetime"])
```

In [ ]:
```
df.dtypes
```

In [ ]:
```
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.w
df.drop(['pickup_datetime'],axis=1,inplace=True)
```

In [ ]:
```
df.weekday = df.weekday.map({'Sunday':0,'Monday':1,'Tuesday':2,'Wednesday
```

In [ ]:
```
fig, ax = plt.subplots(figsize=(13, 13))
corrMatrix = df.corr()
sns.heatmap(corrMatrix, annot=True)
plt.show()
```

In [ ]:
```
from sklearn.model_selection import train_test_split
```

In [ ]:
```
x=df.drop("fare_amount", axis=1)
x
```

In [ ]:
```
y=df["fare_amount"]
```

In [ ]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random
```

In [ ]:
```
x_train.head()
```

In [ ]:
```
x_test.head()
```

In [ ]:
```
y_train.head()
```

In [ ]:
```
y_test.head()
```

In [ ]:
```
print(x_train.shape)
print(x_test.shape)
print(y_test.shape)
print(y_train.shape)
```

In [ ]:
```
from sklearn.linear_model import LinearRegression
lrmodel = LinearRegression()
lrmodel.fit(x_train, y_train)
```

In [ ]:
```
lrmodel_pred = lrmodel.predict(x_test)
```

In [ ]:
```
from sklearn.ensemble import RandomForestRegressor
rfrmodel = RandomForestRegressor(n_estimators = 100, random_state = 101)
rfrmodel.fit(x_train, y_train)
```

In [ ]:
```
rfrmodel_pred = rfrmodel.predict(x_test)
```

In [ ]:
```
from sklearn.metrics import r2_score
lrmodel_r2 = r2_score(y_test, lrmodel_pred)
print("R^2 value for Linear regression is : ", lrmodel_r2)
```

In [ ]:
```
from sklearn.metrics import r2_score
rfrmodel_r2 = r2_score(y_test, rfrmodel_pred)
print("R^2 value for Random Forest regression is : ", rfrmodel_r2)
```

In [ ]:
```
from sklearn.metrics import mean_squared_error
lrmodel_rmse = np.sqrt(mean_squared_error(y_test, lrmodel_pred))
print("RMSE value for Linear regression is : ", lrmodel_rmse)
```

In [ ]:
```
from sklearn.metrics import mean_squared_error
lrmodel_rmse = np.sqrt(mean_squared_error(y_test, rfrmodel_pred))
print("RMSE value for Linear regression is : ", lrmodel_rmse)
```

In [ ]:
```
base = pd.DataFrame()
base["actual"] = y_test
base["predictions"] = lrmodel_pred
base
```

In [ ]:
```
base = pd.DataFrame()
base["actual"] = y_test
base["predictions"] = rfrmodel_pred
base
```

In [ ]:
```
df_pred = pd.DataFrame(rfrmodel_pred)
df_pred
```