Saagnik Sarbadhikari, 50592327
Marcus Hartman, 50398874
Bharath Reddy, 50563984

Project Phase 1

1. Problem Statement (5pts, team)

   Title: Forecasting the Rental Housing Economy of Various Areas Using Machine Learning.

   Problem Statement: Develop a model that can accurately predict rent prices across various locations in the next few years. This will use past data along with future predictions on the economy, which can be used by various clients to find the most cost-effective location to rent.

   Potential of the Project

   Explain the potential of your project to contribute to your problem domain. Discuss why this contribution is crucial?

   Our project aims to be useful in multiple aspects of the world. While our project isn't necessarily predicting the outcome of the economy in the next couple of years, we are aiming to get a gauge of the possibilities that the housing economy could take. This contribution aims to prove its usefulness to both current residents and people looking to move into a new area, as they will have a resource at their disposal that could aide in finding the best-fitting place to live a comfortable life economically.

2. Ask Questions (10pts, individual)

   Saagnik:

   (a) Q1
   (b) Q2

   Marcus:

   (a) Could there be a larger increase in the gap of rent between a high-income state and one that is lower? Could we see people move to other states to save the money, but work remotely?

       This is a significant question as it poses a greater risk ever since the rise of remote work. We have seen people already do this, i.e. moving from New York City to Buffalo for this exact reason. I am wondering if this could be seen on a much larger scale, as rent continues to rise in those higher income states.

   (b) As rent continues to rise, could we see a decline of these prices to combat multi-family apartments?

       The significance of this question arises to illustrate the point of more people opting to live with multiple other people in order to save on rent in smaller buildings. Splitting rent for a three-bedroom apartment between 7 people can be much more affordable than each having their own apartment– will the market react to this? What would this look like as prices keep rising? This is what an additional hypothesis can be formed to question. The best way to answer this question is to keep track of the price differences between one-bedrooms and multi-bedrooms in the future.

   Bharath:

(a) Q1

(b) Q2

3. Data Retrieval (5pts, team)

   To help with individual questions that will be asked in Phase 2 and 3, our group has agreed to focus into two states per person. This will make it easier to strike our own findings, and can find different conclusions using the same model.

   Saagnik:

   Rural:

   Urban:

   Marcus:

   Rural: Minnesota

   Urban: Texas

   Bharath:

   Rural:

   Urban:

Data Retrieval

Sources used (so far):

(a) huduser.gov - 40^{th} Percentile Rents (Historic)

(b) zillow.com - Current housing markets, estimations for an area

(c) rentcast.io - Additional market data for a zip code area

(d) insideairbnb.com - AirBnB rentals by city

Data retrieval consists of a couple calls to certain APIs. While the first two sources are free, rentcast unfortunately charges after the 31^{st} use of the month. Therefore, we only try to do this call once and store the result in the results/ folder we have for this.

The code can be found in the associated ipynb, where the following lines are found:

For huduser.gov:

```
# huduser.gov
huduser_key = open('tokens/huduser').readline()

huduser_MN = requests.get("https://www.huduser.gov/hudapi/public/fmr/statedata/MN",
headers={"Authorization": f"Bearer {huduser_key}"}).json()

huduser_TX = requests.get("https://www.huduser.gov/hudapi/public/fmr/statedata/TX",
headers={"Authorization": f"Bearer {huduser_key}"}).json()
```

For zillow.com:

```
# zillow.com
zillow_rentals = pandas.read_csv("https://files.zillowstatic.com/research/
public_csvs/zori/Metro_zori_uc_sfrcondomfr_sm_month.csv?t=1728278819")
# this is home values forecasts, not rentals. but still poses significance
zillow_forecasts = pandas.read_csv("https://files.zillowstatic.com/research/
public_csvs/zhvf_growth/Metro_zhvf_growth_uc_sfrcondo_tier_0.33_0.67_sm_sa_month.csv?
t=1728278819")
```

For rentcast.com:

```
# rentcast.io
# NOTE: any request using the API key will accumulate towards the monthly limit
(31). please do not go over this

# the commented out part uses the API key you put into tokens/rentcast (req. credit
card info).
# the service is free but still requires credit card info-- if you find something
better use it. update latex and make a little note next to it and we're all good
# rentcast_key = open('tokens/rentcast').readline()

# rentcast_TX = requests.get("https://api.rentcast.io/v1/listings/rental/long-term?
state=TX", headers={"accept": "application/json", "X-Api-Key": rentcast_key})
# f = open("results/rentcast_TX", "w")
# f.write(json.dumps(rentcast_TX.json()))
# f.close()

# rentcast_MN = requests.get("https://api.rentcast.io/v1/listings/rental/long-term?
state=MN", headers={"accept": "application/json", "X-Api-Key": rentcast_key})
# f = open("results/rentcast_MN", "w")
# f.write(json.dumps(rentcast_MN.json()))
# f.close()

# loads data stored from api calls above. costs money if we go above 31 calls / month
rentcast_TX = json.loads(open('results/rentcast_TX').readline())
rentcast_MN = json.loads(open('results/rentcast_MN').readline())
```

For insideairbnb.com:

```
# airbnb
airbnb_TX_data = pandas.read_csv("https://data.insideairbnb.com/united-states/tx/dallas/
2024-08-17/visualisations/listings.csv")
airbnb_MN_data = pandas.read_csv("https://data.insideairbnb.com/united-states/mn/
twin-cities-msa/2024-06-24/visualisations/listings.csv")
```

4. Data Cleaning (10pts, team)

   For each dataset, we cleaned up all of the NaNs, converted every dataset into a similar JSON, and dropped any unnecessary columns pertaining to our problem. Each are separated out by their source.

For huduser.gov:

```
# 1. cleaning hudusers.gov of unnecessary columns
def cleanCounties(countyInfo):
    columnsToDrop = ['town_name', 'metro_name', 'fips_code', 'FMR Percentile',
    'metro_name', 'statename', 'statecode', 'smallarea_status']

    for county in countyInfo:
        for n in columnsToDrop:
            if n in county:
                county.pop(n)

    return countyInfo


countyInfo_MN = cleanCounties(huduser_MN['data']['counties'])
countyInfo_TX = cleanCounties(huduser_MN['data']['counties'])
```

For zillow.com:

```
#2. Zillow rentals: drop unnecessary columns
zillow_rentals.drop(["RegionID", "SizeRank", "RegionType"], axis=1, inplace=True,
errors='ignore')

#3. Zillow rentals: drop all that aren't for MN, TX
zillowRentals_MN = zillow_rentals.query('RegionName.str.contains("MN")')
zillowRentals_TX = zillow_rentals.query('RegionName.str.contains("TX")')

#4. Zillow forecasts: drop unnecessary columns
zillow_forecasts.drop(["RegionID", "SizeRank", "RegionType", "BaseDate"], axis=1,
inplace=True, errors='ignore')

#5. Zillow forecasts: drop all that aren't for MN, TX
zillowForecasts_MN = zillow_forecasts.query('StateName == "MN"')
zillowForecasts_TX = zillow_forecasts.query('StateName == "TX"')
```

For rentcast.com:

```python
#6. rentCast: drop unnecessary columns
def cleanRentals(rentalInfo):
    columnsToDrop = ['id', 'formattedAddress', 'addressLine1', 'addressLine2',
    'latitude', 'longitude', 'status', 'listingType', 'listedDate', 'removedDate',
    'createdDate', 'lastSeenDate', 'daysOnMarket', 'yearBuilt', 'history']

    for rental in rentalInfo:
        for c in columnsToDrop:
            if c in rental:
                rental.pop(c)

try:
    cleanRentals(rentcast_MN_json)
    cleanRentals(rentcast_TX_json)
except TypeError: # cell already ran
    pass

#7. rentCast: sort data by county, store in dictionary for easier lookup
def convertToDict(rentalInfo):
    res = dict()
    for rental in rentalInfo:
        county = rental['county']
        if county not in res:
            res[county] = []
        res[county].append(rental)
    return res
```

For insideairbnb.com:

```python
#8. airbnb: get rid of unnecessary columns
airbnb_MN_data.drop(["id", "name", "host_name", "host_id", "neighbourhood_group",
"latitude", "longitude", "last_review", "reviews_per_month", "number_of_reviews_ltm",
"license"], axis=1, inplace=True, errors='ignore')
airbnb_TX_data.drop(["id", "name", "host_name", "host_id", "neighbourhood_group",
"latitude", "longitude", "last_review", "reviews_per_month", "number_of_reviews_ltm",
"license"], axis=1, inplace=True, errors='ignore')

#9. airbnb: get rid of duplicates
airbnb_MN_data.drop_duplicates()
airbnb_TX_data.drop_duplicates()

#9. airbnb: get rid of rows where price is not available
airbnb_MN_data.dropna(subset=['price'], how='all', inplace=True)
airbnb_TX_data.dropna(subset=['price'], how='all', inplace=True)

#10. airbnb: sort by room_type, convert to dictionary
def convertToDict_BNB(df : pandas.DataFrame):
    res = dict()
    for _, row in df.iterrows():
        roomType = row['room_type']
        if roomType not in res:
            res[roomType] = []
        res[roomType].append(row)
    return res

airbnb_MN = convertToDict_BNB(airbnb_MN_data)
airbnb_TX = convertToDict_BNB(airbnb_TX_data)
```

5. Exploratory Data Analysis (20pts, individual)