Analysis of Vacation Rental Metrics and Dynamic Pricing Strategies Using Reinforcement Learning

Cole Beasley
Computer Science, NYUAD
crb596@nyu.edu

Muhammad Anas
Computer Science, NYUAD
muhammad.anas@nyu.edu

Advised by: Azza Abouzied, Christina Pöpper

ABSTRACT

Competitive pricing is one of the key factors to success in a vacation rental host's ability to attract visitors and therefore generate revenue. The projected price and expected revenue is also a metric that anyone considering entering the market would like to know before making any significant investment into the listing of a home.

This project covers the development of a deep reinforcement learning model that allows suggestive optimal pricing for a vacation rental property. Based on the historical data of property rentals, including the given property features, nearby listings analysis, and external features influencing rental prices, the model predicts revenue-maximizing dynamic pricing for the property host.

The model's results have been meshed with market analysis and delivered in an interactive toolkit that the property owners can apply to their personal property's context. In the interactive dashboard, market information is presented against property-specific features to contextualize the information and help users make data-backed decisions.

KEYWORDS

vacation rental, Airbnb, VRBO, dynamic pricing, deep reinforcement learning, data visualization

Reference Format:
Cole Beasley and Muhammad Anas. 2023. Analysis of Vacation Rental Metrics and Dynamic Pricing Strategies Using Reinforcement Learning. In *NYUAD Capstone Seminar Reports, Spring 2023, Abu Dhabi, UAE.* 9 pages.

1 INTRODUCTION

As the vacation rental market continues to increase in competitiveness, it can be challenging for homeowners and potential hosts to competitively price their rentals on platforms such as Airbnb or VRBO. It can be challenging to gauge the market due to the numerous factors that affect tourism, and on top of that, hard to optimize the trade off between maximizing nightly rates against occupancy. By empowering users with access to accurate, up-to-date market data, consideration of property features and how they change the value, and how the location of a listing makes it more or less competitive, better decisions regarding their management strategies can be made. While the travel industry has been typically dominated by hotel corporations with access to in-depth market information, it is becoming increasingly common globally for individuals to be making these decisions with large personal investments.

To help solve this problem, a tool was conceived which uses live data from the short-term rental market and a reinforcement learning model to make accurate predictions about the optimal price for a given property. This tool will be used by homeowners who want to rent out their properties as vacation rentals, as well as by prospective property buyers who want to gain insight into the potential profit that can be made from a property before investing. By using live market data and a deep reinforcement learning model, this tool can make more accurate predictions than traditional pricing methods. It will take into account factors such as location, property type, and amenities, as well as real-time information about supply and demand in the market. This allows it to provide users with up-to-date, reliable pricing information that can help them make informed decisions about how their unit will perform. In addition to providing users with the ability to set competitive prices for their properties, our tool also offers interactive visualizations for hosts to see how their property fits in against the market. Through these two means, users can make decisions informed by data and have a better understanding of how they can expect investment to perform.

2 RELATED WORKS

While there has been limited research into how dynamic pricing can be generated based on user inputs for vacation rentals, there is existing literature and examples that incorporate the elements outlined to be encompassed by this project.

2.1 Dynamic Pricing

Dynamic pricing involves setting prices for products in real-time based on market conditions, where DRL algorithms have been shown to be effective at learning the optimal pricing strategies for different products. The key advantage of using deep reinforcement learning for dynamic pricing is its ability to adapt to changing market conditions and customer behavior as it progresses as a time series, rather than just a singular state.

In the paper "Real-time Dynamic Pricing in a Non-Stationary Environment Using Model-Free Reinforcement Learning," the authors demonstrate how a model-free reinforcement learning algorithm can adapt to changing conditions and make optimal pricing decisions in real-time.[6] The Q-learning with eligibility trace, Q(), algorithm was used to solve the non-stationary Markov decision process, and it converges and produces a better policy than the standard Q-learning algorithm. Furthermore, it also integrates a demand function which could either be provided or the model adjusts itself to, which helps in deriving a policy to map out the market demand model.

Another important benefit of using deep reinforcement learning for dynamic pricing could be its ability to incorporate fairness constraints into pricing decisions. In the paper "Reinforcement Learning for Fair Dynamic Pricing," the authors test their algorithm on a simulated environment and show that it is able to learn optimal pricing strategies while still maintaining fairness.[4] By using deep reinforcement learning to promote fairness in pricing, businesses can help to create a more equitable market.

Another common usage of DRL is to develop stock trading platforms, where algorithms are trained to make buy-and-sell decisions based on historical data and real-time market information. The paper "Model-based Deep Reinforcement Learning for Dynamic Portfolio Optimization," proposes a DRL architecture for autonomous trading to make optimum investment decisions based on a set global objective.[9] It utilizes several prediction, data augmentation, and behavior cloning modules to develop on or off-policy RL agents that use historical financial market data to simulate a trading environment that is profitable and risk-sensitive.

2.2 Data Visualization

The issue of providing users with information regarding an asset in a specific market is not novel, and there are several approaches that have been taken to approach this problem. Real Estate companies like Zillow for example, employ techniques including data layers on a map to contextualize geospatial data.[10] The site classic.com, indexes car auctions to show market trends in the classic car industry.[3] Both of these tools are good examples of how a lot of data can be concisely brought down to a couple of key metrics and communicated to users without losing the depth and richness of the data points.

One thing neither of these examples does however, is provide users much interactivity to see what sort of decisions users should be making with the data being presented. New York Times provides a nice tool that allows users to input a set of parameters about a property, and show whether it is a better choice to rent or buy a property.[2] The approach utilizes user input and shows how direct changes in the input values affects the output.

3 METHODOLOGY

3.1 Data Collection

One of the key challenges in developing a tool using reinforcement learning for price predictions is collecting accurate data to feed the environment in the training process. This data is essential for making precise predictions about the optimal price for a given property, as it allows the agent to take into account factors such as location, property type, and amenities, as well as real-time information about supply and demand in the market. To inform users about the market and general trends beyond price predictions from the RL agent, data points are also required to be quickly queryable for access by the dashboard.

The main approach to the project's collecting of data is through data scraping, which involves using packages such as Python's Beautiful Soup library, to retrieve information about whether listings on sites like Airbnb are available on a given date.

The scraper works in two modes, exploration, and revisitation. In the exploration mode, the scraper is visiting short-term rental marketplaces including Airbnb and VRBO, and going through properties that are advertised on the map methodically to save their information. Through this process, the features of a property are processed and an ID is saved so that the property can be returned to in the future. In the revisitation phase of data scraping, the scraper goes through previously recorded properties and visits their specific pages. Looking at the availability information, the scraper saves in bookings including their start and end days, when this booking was first recorded by the scraper, and the price of the nights immediately surrounding the booking. Some information is unfortunately lost through this method, such as exactly when the booking was made, if a period of blocked dates is a singular or multiple back-to-back bookings, and the exact pricing information that was used when this booking was made. That being said, as no other official way exists to access the data, this is the method most commonly employed in similar projects, and it gives good enough results for high-level market analysis and training of models.

Making use of publicly available data through data scraping is a common approach that many other projects in this field employ as mentioned. Scalability can be costly due to the need to use rotating proxies to bypass mechanisms put in place to protect online rental marketplace sites from attack, however. To supplement the information that was gathered throughout the capstone, and provide more in-depth historical data, some precompiled datasets were utilized. Inside Airbnb is a research group and data company that offers Airbnb datasets.[1] Utilizing their New York data, we were able to supplement our scraped data and extend our history back to December 2021.

To manage the property information and hundreds of booking data points associated with each listing, a database that could handle geospatial queries was utilized. MongoDB was chosen to fill this role due to its generous free tier and ability to index GeoJSON objects. Two collections were set up, one that stores property information indexed on location using MongoDB's geosphere index, and another that stores property booking information and a reference to the property information in the first collection. This methodology allows for quick spatial queries such as 'find all properties within 1km of these coordinates', which is useful for the mapping functionality of the dashboard.

As this data collection mechanism proved effective, it could be scaled to much bigger regions or global datasets with sufficient resources. The queries on property data are quick despite the dataset size and could be scaled to fit a much larger range of properties. With the main limitations in scalability being database size, scraping throughput, and time, these limitations could be addressed with further investment in the currently utilized services.

3.2 Deep Reinforcement Learning Approach

This project utilizes a deep reinforcement learning (DRL) approach, which combines deep neural networks and reinforcement learning (RL) algorithms to learn complex decision-making policies. In DRL, an agent interacts with an environment to optimize a task (or decisions) by taking certain actions and receiving corresponding feedback in the form of rewards. The objective of the agent is to maximize the cumulative reward over time by learning an optimal policy.

To apply DRL in our real-world case of the short-term rental market, it is essential to have an environment that simulates the actual market. This environment is set up using deep learning models that accurately represent the dynamics and trends of the market. By interacting with this simulation, the naive agent can experiment with various strategies and permutations of actions to eventually train over an optimal decision-making policy. This decision-making policy should output the dynamic pricing strategy over a year for an input property that would help maximize its revenue based on the RL agent's learning.

3.3 Environment Development

The environment serves as a simulatory model for the short-term rental market. This allows the agent to interact by providing a specific current state and action and receive a corresponding reward. The accuracy of the environment is essential as that sets the baseline and models the reality upon which the RL agent learns its policy.

The environment models the booking status of a property based on various input attributes, including a given price and date. To achieve this, a neural network model is trained using historical records of Airbnb features and their booking statuses to predict the likelihood of a property being booked on a specific day at a given price. The learnings of the market dynamics are utilized in the simulatory environment.

A customized environment to model the Airbnb market was developed according to the OpenAI gym API configurations for RL simulations. This allows the standard RL libraries like OpenAI's stable-baselines3 to train different policies on the environment. These are the main components of the environment:

Observation Space: The observation space comprises all the pertinent features of a property. Each observation instance corresponds to the characteristics of an Airbnb property, alongside the price and given date. The observation space provides us an instance of a current state that our environment is in, and the next state that it will transition to upon an undertaken action. The attributes, comprising the property and host characteristics, set price, and date (decomposed into day of week, month, year, and is_holiday) that comprise the observation space are provided in Table 1.

Action Space: The action is essentially the price that is set for a property (state) on a given day. The action space in this environment is the normalized price distribution ranging from 0 to 1. It allows the RL agent to "select" an appropriate price for the property in consideration.

Reward: The reward is the weighted average revenue that could be expected for a property state on a given day at the

host_is_superhost	accommodates	bathrooms
bedrooms	beds	review_cleanliness
review_rating	review_location	review_value
Neighbourhood	is_holiday	Day_Monday
Day_Tuesday	...	Day_Sunday
Month_1	...	Month_12
Year_2021	Year_2022	Year_2023

Table 1: Observation Features

given price (action). The reward system uses the booking confidence provided by the neural network to model the revenue. If the booking likelihood is over a 0.85 threshold, then the property is assumed to be booked and the reward is the price action for the day. Similarly, below the threshold of 0.15, it is considered to be unbooked, producing 0 reward for the day. For other values, reward is returned as a weighted average following the equation below:

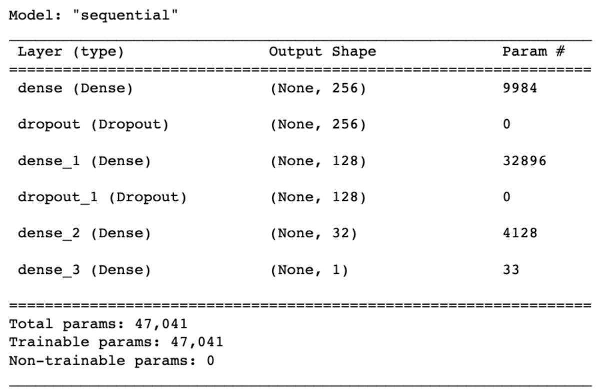
```
chance = random.uniform(0, 1)
if chance < bookingConfidence:
    return (price + (price*bookingConfidence))/2
return (0 + (price*bookingConfidence))/2
```

This helps model the real-world randomness as well as provide a more fair average expected revenue that the RL agent can use to optimize its decision-making

Neural Network: The neural network is designed to discern trends in rental property bookings across the year. The model is developed to capture the complex relationships between the property attributes and the market trends, accounting for the variations in host characteristics such as review ratings, property information of location or the number of rooms, price differences, and seasonality demand across the year. This deep learning approach allows for the development of a more realistic environment to predict booking probability, which can better reflect real-world situations than theoretical models can.

The model is trained on a dataset of about 35,000 properties for their daily booking status across 2021 to 2023. This provides 5 million instances of booking records that the model learns to classify on. Each data input comprises 38 attribute features: 10 corresponding to the property characteristics; 5 categorical variables for the neighborhood (corresponding to 5 boroughs); and 23 variables that decompose the date into the day of the week, month of the year, the year itself and if its a holiday.

The model is trained over 200 epochs until the model converges to a training accuracy of around 0.706. The batch size is set to 128 to facilitate a balance between the risk of overfitting as well as convergence to a more stable and accurate gradient estimation as it is computed on sufficient samples in each iteration. The NN follows the following architecture in Figure 1.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 256)               9984

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 128)               32896

 dropout_1 (Dropout)         (None, 128)               0

 dense_2 (Dense)             (None, 32)                4128

 dense_3 (Dense)             (None, 1)                 33

=================================================================
Total params: 47,041
Trainable params: 47,041
Non-trainable params: 0
_________________________________________________________________
```

Figure 1: NN Architechture

The architecture is intended to model the complexities in the data and it does with expanding the incoming 38 features into 256 neurons. The idea is to have substantial parameters to train the weights for the multiple input features and the large dataset of 5 million samples. The model also uses Dropout layers between the intermediate layers to regularize the model. Since the model output (booking of a property) is noisy and heavily randomized, the use of dropout layers helps prevent overfitting and enhances the generalization capabilities of the model over variations in the input data.[8] It randomly deactivates 20% of the neurons in each layer, making the network less sensitive to precise details of individual neurons.

While the intermediate layers are activated by the "ReLU" activation, standard for its performance and efficiency, the final output layer uses "sigmoid" activation to model the binary classification based on the "binary_crossentropy" loss. In this binary classification problem, the sigmoid function converts the weighted sum of inputs into a probability value that represents the likelihood of the input being classified as "booked".

The hyperparameters for the model training are selected after evaluation of the model's performance for different combinations of hyperparameters.

3.4 Reinforcement Learning Agent

Reinforcement learning enables an agent to acquire an optimal decision-making policy that maximizes rewards by interacting with a simulated environment. The goal of this project is to apply RL to learn a dynamic pricing strategy that maximizes revenue for a specific property in the short-term rental market.

The agent uses trial-and-error exploration to determine the best action (price) for a given state (property) based on feedback received from rewards (revenue). By converging to an optimal policy, the agent learns to set prices that result in higher revenues over time, starting with exploration and eventually exploiting its preferences. Various RL policy algorithms approach optimal policy learning differently. Two relevant algorithms explored in this project are discussed below.

DQN: Deep Q-Network is the most popular RL algorithm that combines simplistic Q-learning with deep neural network. The approach uses Q-value as an action-value function that maps states to the expected future rewards of taking each possible action on it. The agent learns to optimize its policy by iteratively updating its Q-values using the Bellman equation.[5] The deep neural network helps with approximating the Q-value function, which allows the agent to generalize its knowledge of the environment to unseen states. DQN is suitable where the state space is large and complex, and the action space is discrete. In our implementation, DQN is used to provide the action values and the corresponding anticipated reward for each action, where the action is the change in price to the existing price. For an example of a price set to 200, it could explore what the rewards could be by changing the prices by either -10, 0, or 10, and incorporate the learnings received. Because of DQN's limitation to discrete action space, which also causes overestimation of Q-values, large updates, and unstable convergence, it leads to suboptimal policies.

PPO: Proximal Policy Optimization algorithm is better suited to problems where the action space is continuous. Instead of the action being the "change in price", the value of the price itself, chosen from a Gaussian distribution, becomes the action. For example, the policy itself chooses a price, such as 210 or 190 as the action to take. PPO uses a deep neural network to approximate the policy function and learns the policies directly. The policy function maps states to actions and determines how an agent should behave in an environment. PPO uses a surrogate objective function that constrains the size of policy updates to prevent large policy changes that could lead to instability. As a result, PPO is known to be more stable than other policy gradient methods, and each iterative learning is in small increments.[7] This is essential in our case to gradually learn the accurate pricing values rather than the fixed changes in prices as actions.

The PPO approach is utilized for the implementation because of its performance and suitability to the use-case. Once the custom Gym API environment is set and the stable-baselines3 model is initiated and linked to the environment, the RL Agent undertakes the following major steps through its PPO policy to train the model and learn the optimal pricing strategy over time.

- **Reset:** The RL agent picks a random property from the training set. This is the reset of the environment, where it begins with a new property for the new current state.
- **Steps:** The agent then takes a series of steps on the current property. The steps consist of an agent suggesting a property price as an action. Setting this price for the property on the given day, the model tries to estimate what the expected revenue could be. This revenue is provided from the reward function explained above.
- **Intermediate iterations:** This process is repeated 10 times for each property on a given day so that the PPO algorithms can choose multiple price guesses and learn from the experience how each price reflects a change in revenue.
- **Time iterations:** Each property state iterates through the 365 days of the year. Hence, the model makes this price guess 10 times for each day and does this for all the 365 days of the year for each property to learn the dynamic pricing. In total, the model trains for 3,650 timesteps per property.
- **Store results:** Each iteration records the current state, the action taken, corresponding reward feedback, and the next state this leads to. The agent learns on each time step on how a particular action taken on a particular state results in the subsequent change in reward. The mapping of these results allows it to update its weights accordingly.

The Agent, while training, repeats the above processes for all the properties. Each epoch trains for a set of 60 random properties. With each property training for 3,650 timesteps, this totals up to 219,000 timesteps (or iterations across steps) in each epoch. The model is trained for a total of 100 epochs for this project, totalling to a 21.9 Million timesteps. Overtime, the model should evolve from exploratory decision-making, where it makes random price decisions, to policy exploitation where it converges on learned and aware price decision-making.

3.5 Data Visualizations

One of the key elements of the project, and what users will ultimately interact with, is the dashboard. The deliverance of trends within the data, as well as the key results from the reinforcement learning model through the interactive dashboard, focuses the project on user decision-making. With the overarching goal of helping vacation rental property owners make data-informed decisions, the dashboard serves as the main conduit between data, and the ability for users to make choices regarding their property management. Decisions such as pricing strategies, how aspects of the property affect its value and likelihood to be booked, and ROI all aim to be addressed through the dashboard.

With the aforementioned MongoDB setup, it was decided that the popular MERN web stack (Mongo, Express, ReactJS, NodeJS) would be best suited for the development of this web app. With extensive capabilities from supporting libraries including D3 and MaterialUI, React was the best approach to develop powerful dashboards that could be fully tailored to fit the project design requirements.

There are two main components of the dashboard. Upon loading into the webpage, the user is greeted with the market analysis section, where users can explore the current market through both spatial and temporal visualizations. Users can then switch over to the price intelligence section of the dashboard, where more personalized pricing strategies are visualized with data coming from the RL agent.

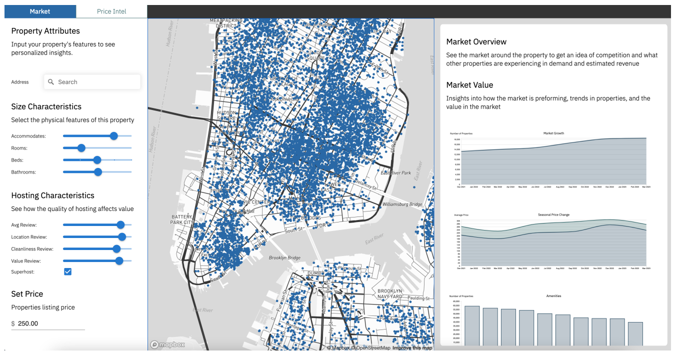

Market
Price Intel
Property Attributes
Input your property's features to see personalized insights.
Address
Search
Size Characteristics
Select the physical features of this property
Accommodates:
Rooms:
Beds:
Bathrooms:
Hosting Characteristics
See how the quality of hosting affects value
Avg Review:
Location Review:
Cleanliness Review:
Value Review:
Superhost:
Set Price
Properties listing price
$ 250.00
Market Overview
See the market around the property to get an idea of competition and what other properties are experiencing in demand and estimated revenue
Market Value
Insights into how the market is preforming, trends in properties, and the value in the market
Market Growth
Seasonal Price Change
Amenities
Brooklyn Bridge
Williamsburg Bridge
BROOKLYN NAVY YARD
BATTERY PARK CITY

Figure 2: Market overview on the dashboard

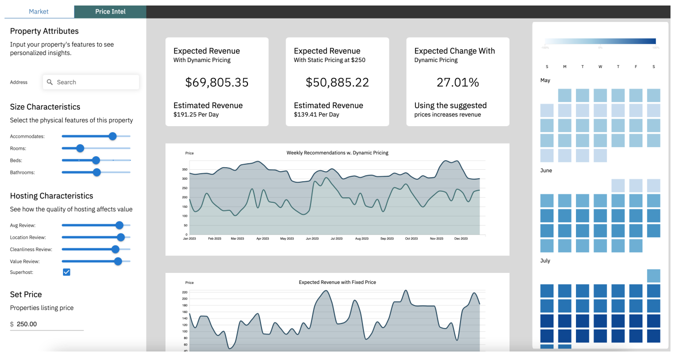


Figure 3: Price suggestions from the dashboard

In either view, the tool shows an input panel for users to enter property information. The objective is for this tool to show feedback about any hypothetical or veritable property, and therefore any combination of inputs can provide meaningful feedback. Inputs start grayed out and at market-average, and then can be fine-tuned to be as hyper-specific to any property as the user wants. As users input information about their property including location, size, rental-specific characteristics such as reviews, and their nightly rate, the graphs reflect how particular combinations of property features affect performance in the market compared to the market average. For example, a user can adjust the number of beds as an input, and see how the average property price for listings of that size has shifted over time, and how the expected revenue increases or decreases. This allows users to see how decisions such as creating a bedroom, or buying a different size property, can have different returns.

In the market view section, an interactive map was developed to give context to what competition exists. A Mapbox map was chosen due to its highly customizable interface via Mapbox Studio, and its support for GeoJSON data layers meaning thousands of points can be plotted without severe performance loss. The market view also includes custom D3 graphs showing how the market changes over time and how it can be categorized. This breakdown of the complex market space into a few simple graphs shows where a property would fit among thousands of diverse listings.

The price intelligence view switches from the analytic overview of the current market to forecasting how a property could do given a set of parameters. Utilizing calls to the RL model, information is presented about how pricing strategies can optimize revenue, and how booking likelihood is expected to look over the year. This view aims to show users how their decisions will shift how the property performs and can adhere to their personal preferences. For example, if a user wants a more aggressive strategy, then they can up their base price and see how occupancy likelihood shifts down, but the reward from bookings that do occur increases. By letting users see how their decisions affect price forecasting, they can make personal decisions on what they are looking to get out of their property.

4 RESULTS

The results of the deep learning environment and reinforcement learning agent help analyze the performance of the solution. Providing the final output, they suggest how useful DRL could be in price optimization. The performance metrics and results of both the environment Neural Network and RL Model are discussed in the following sections.

4.1 Environment Performance

The environment's accuracy and reliability are critical as it provides the real-world market baseline simulation for the agent to train on. The RL Agent is as good as the environment itself.

The binary classifier NN predicts the probability that the property is booked at the given price on a particular day. Table 3 provides the performance metrics for the binary classifier over training and test datasets.

Table 2: Model Performance Metrics

	Metrics				
	AUC	Accuracy	Recall	Precision	Prevalence
Training	0.785	0.709	0.722	0.630	0.414
Test	0.785	0.709	0.722	0.630	0.414

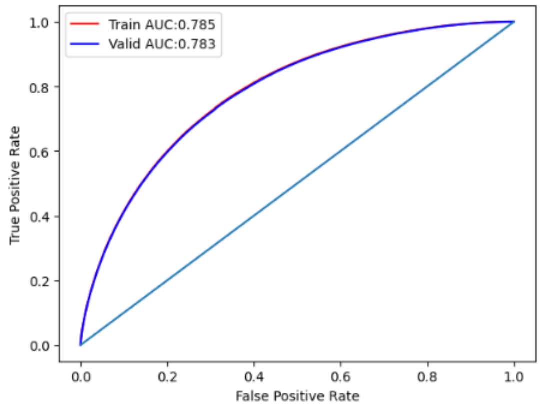


Figure 4: ROC Curve

While the accuracy of 0.707 doesn't reflect an ideal classifier, predicting the booking status for a specific date of the year is complex and heavily influenced by randomness and noise. Therefore, classifying about 70% of bookings correctly proves substantial for our use case. The area under the ROC curve scores for the classifier are close to 0.8, suggesting that the classifier is doing a decent job of separating the two classes. Furthermore, the ROC curve in Fig. 4 visualizes the substantial performance of the true positive rate against the false positive rate over the threshold, the values for which are provided in the confusion matrix at the threshold of 0.45 in Table 3.

Table 3: Confusion Matrix

True Negative	False Negative	False Positive	True Positive
43.5%	14.2%	15.1%	27.2%

The classifier's main purpose is to identify the trends in booking probability against price and day of the year. Table 4 provides the characteristic features of a test property, originally listed at $348.0. Figures 5 and 6 illustrate the model's performance in predicting the booking probability across the price range and time of the year for the sample property.

Figure 5: The model provides a fair approximation of how increasing the price lowers the probability of the property being booked, indicating clear thresholds around which it's unlikely to book this property. In this case, there's a drastic fall when increasing the price above the $348.0 mark that the owner set.

Table 4: Feature Variables

Feature	Value
is_superhost	1
accommodates	5
bathrooms	1
bedrooms	3
beds	3
review_scores_cleanliness	4.73
review_scores_rating	4.82
review_scores_location	4.86
review_scores_value	4.79
price	348.0
neighbourhood	Manhattan

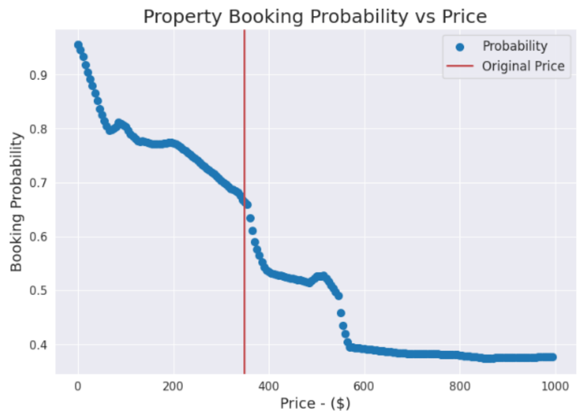


Figure 5: Booking Probability vs Price

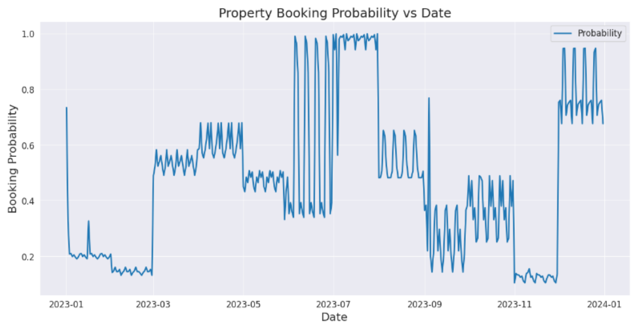


Figure 6: Booking Probability vs Date

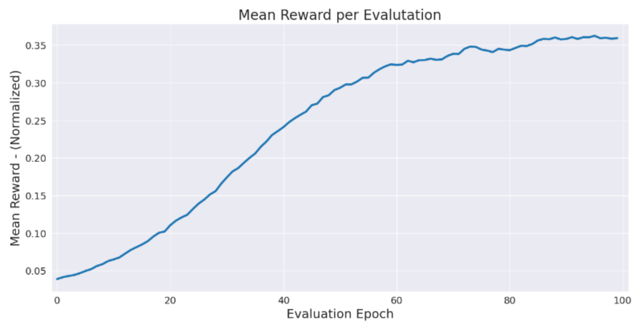


Figure 7: Mean Reward per Evaluation

Figure 6: The NN critically understands the time and seasonality features across the year and provides valuable insights into when demand for the property is expected to be high or low. The graph provides some notable insights:

- Demand changes across the week and weekends have a higher booking probability.
- Months such as Jan, Feb, and November, have very low booking projections, whereas summer and December holiday seasons are popular times. For a property based in Manhattan, this is very well representative of the tourism trends.
- National holidays have critical effects: 1st Jan sees high and July 4th sees low demand for this property

4.2 RL Agent Performance

The aim of the RL agent is to maximize its reward function and corresponding revenue. The RL agent's performance is evaluated over the 100 epochs it trains through, and in each evaluation, it is tested against a set of 200 validation properties. The mean normalized reward earned over the properties represents the agent's learning performance, denoted as the daily revenue earned per property for the year, normalized to its original price. The convergence of the reward indicates that the model is adequately trained to provide meaningful results. Figure 7 displays the RL agent's performance over 100 epochs, illustrating how an untrained model making random predictions performs poorly at first but eventually learns to converge towards an optimal strategy. The last 30 epoch runs show no significant increase in the mean average revenue per property.

The trained RL model is tested against the same property in Table 3. The Agent outputs a weekly pricing plan for the entire year 2023 and the corresponding revenue expected. Figure 8 visualizes the results, suggesting a dynamic pricing strategy that hovers above $400 for the property initially priced at a static rate of $348. The model predicts that the owner could expect the following annual revenue if they adopt this dynamic pricing policy:

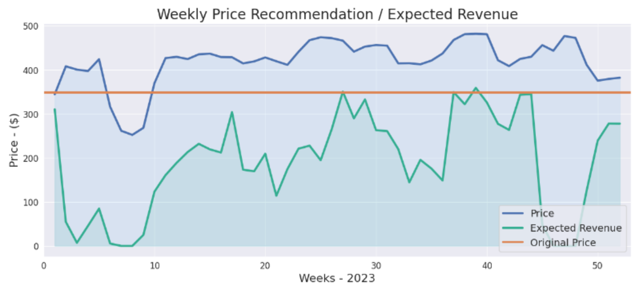


Figure 8: Week Price Recommendation & Revenue

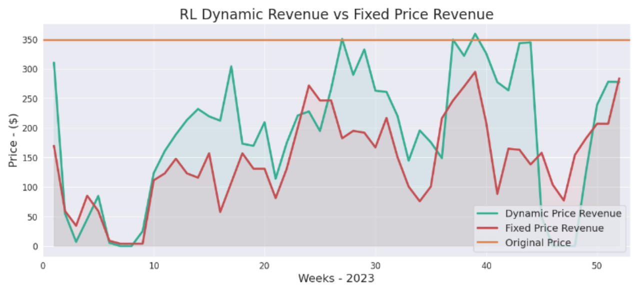


Figure 9: Dynamic Revenue vs Fixed Price Revenue

Annual Revenue: $69, 182.39

Comparatively, testing the model against the case where the owner continues to list the property at the static rate of $348 reveals that for most of the year, the dynamic pricing strategy would generate higher revenue. Figure 9 presents a comparison between the expected revenue for the case of suggestive dynamic pricing strategy versus the initial static pricing for all the weeks of the current year, 2023, showing that the expected annual revenue for the static price is:

Annual Revenue: $52, 866.84

Thus, a user could expect about **30% more revenue** if following the dynamic pricing strategy recommended by the system.

The above analysis is conducted for a set of 200 validation properties, where the model predicts the anticipated revenues for the entire year if the user follows the dynamic pricing policy versus if they set their original static pricing policy. The total annual revenue for these properties with the dynamic pricing policy was $7.1 million, compared to $5.5 million for their original individual prices.

This represents a 28.2% increase in revenue using the DRL predictive pricing for 200 test cases.

5 FUTURE DEVELOPMENT AND CONCLUSIONS

A lot has been developed within the scope of this project, and a working proof of concept has successfully been created.

To take this project to the next step, where deployment to users takes place, a few steps need to be undertaken.

The first step is scaling data collection to broader regions. Right now only the greater New York City area is covered by the model and dataset, and more expansive data collection could be done to make the tool more universal. There is also room for optimization within the model. Right now price predictions take roughly one to three minutes to generate, which is far too long for an integrated use case. To serve virtually live predictions as users change sliders, it will be key to optimize how the model is generating its outputs. The dashboard layout itself also can be optimized. Looking at other design examples, the dashboard could be more user-friendly in how it delivers information. State management is something the dashboard could also improve on to optimize performance and load times, and minimize the number of API requests necessary to populate the dashboard.

Once these development points are addressed, other features could be added to make the tool even more useful and prevalent to hosts. For example, a user could link their property through the Airbnb API, and the model could take into account live property-specific behavior with a higher weight to make price predictions more accurate.

In its current state, the project serves as a solid MVP and demonstrates how synthesizing large amounts of data in complex markets can be beneficial. With its interactivity, this project transforms data into dynamic predictions, and then to visualizations, all within the framework a user outlines with their personal preferences and property details.

ACKNOWLEDGMENTS

We would like to thank Professor Azza Abbouzeid for advising us on this capstone project to help us develop the DRL Agent and end product. We also thank Anh Mai for his constant guidance on approaching the problem and Reinforcement Learning concepts. We also extend thanks to the NYUAD HPC Jubail cluster team who helped us with using their computing resources.

REFERENCES

[1] Inside Airbnb. [n.d.]. http://insideairbnb.com/get-the-data/

[2] Mike Bostock, Shan Carter, and Archie Tse. 2014. Is it better to rent or buy? https://www.nytimes.com/interactive/2014/upshot/buy-rent-calculator.html

[3] Classic.com. [n.d.]. https://www.classic.com/

[4] Roberto Maestre, Juan Duque, Alberto Rubio, and Juan Arevalo. 2018. Reinforcement learning for Fair Dynamic Pricing. *Advances in Intelligent Systems and Computing* (2018), 120–135. https://doi.org/10.1007/978-3-030-01054-6_8

[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[6] Rupal Rana and Fernando S. Oliveira. 2014. Real-time dynamic pricing in a non-stationary environment using model-free reinforcement learning. *Omega* 47 (2014), 116–126. https://doi.org/10.1016/j.omega.2013.10.004

[7] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[8] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[9] Pengqian Yu, Joon Sern Lee, Ilya Kulyatin, Zekun Shi, and Sakyasingha Dasgupta. 2019. Model-based Deep Reinforcement Learning for Dynamic Portfolio Optimization. https://doi.org/10.48550/ARXIV.1901.08740

[10] Zillow. [n.d.]. https://www.zillow.com/homes/new-york-city_rb/