Artificial Neural Networks Final Project

Assis. Prof. Dr. Onur ERGEN

REAL ESTATE MARKET PREDICTION

DÜNDAR EMRE ÖZBIRECIKLI

Date of the Deadline: January 21st 23.59pm

Github:
https://github.com/dundaremreozbirecikli/ANN-Real-Estate-Price-Prediction

1. Abstract

Real estate market is growing due to the needs of human and it is becoming not easy to predict the future prices of the properties in the current economic system. Prediction of the real estate market of Madrid, Spain is done in this project by using the Artificial Neural Networks and dataset that includes numerous features that effects real estate prices. Each step of the project will be covered in this report which are analyzing the dataset, selecting features, data preprocessing, training, testing and predictions.

2. Introduction

The solution to the problem of predicting real estate prices accurately comes with choosing correct features from the obtained dataset [1]. In this project, Madrid is selected for predicting the real estate market. Preprocessing should be applied to the dataset effectively to remove the unnecessary features to train from the dataset and applying necessary transformation, encoding operations. One-hot encoding is used for categorical features in this project because neural network requires numeric inputs. Sequential model is selected and the model is compiled with the "Adam" optimizer and mean squared error is used as loss function which is effective for regression. In the training, data is splitted as 70% train and 30% test data. After those predictions are made and results are shown as plots and comparison.

Used libraries in the project:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, KFold,
cross_val_score
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.decomposition import TruncatedSVD
from keras.models import Sequential
from keras.layers import Dense, Dropout, Input
from keras.regularizers import l2
from keras.callbacks import EarlyStopping
```

3. *Dataset and Preprocessing*

- Dataset: Dataset consist of 21742 rows and 58 columns. Data dictionary is shown in the table below. 'Unnamed: 0', 'id', 'title', 'latitude', 'longitude', 'portal', 'door', 'rent_price_by_area' variables are dropped from the data that used in the project.

Variable Name	Description
ID	Identifier ID
Title	Title from listing
Subtitle	Neighborhood and city
sq_mt_built	Square meter built
sq_mt_useful	Square meter useful
n_rooms	Number of rooms
n_bathrooms	Number of bathrooms
sq_mt_allotment	Square meter allotment
latitude, longitude	Latitude, Longitude
raw_address	Address
is_exact_address_hidden	Boolean values
buy_price	Target Value

Figure 3.1 - Madrid Real Estate Dataset Dictionary

Dataset also includes the features of the house that effects its price like room number, floor number, street name, built year, has lift or not, has a balcony or terrace etc. Small portion of the dataset is shown in the Figure 3.2.

	id	title	subtitle	sq_mt_bu	sq_mt_us	n_rooms	n_bathroo	n_floors	sq_mt_all	latitude	longitude	raw_addre	is_exact_a	street_nar	street_nur	portal	floor	is_floor_u	door	neighborh	operation	rent_price	rent_price	is_rent_pr	buy_price	buy_price_
0	21742	Piso en ve	San CristÃ	64	60	2	1					Calle de G	FALSE	Calle de G	64		3	FALSE		Neighborh	sale	471		FALSE	85000	1328
1	21741	Piso en ve	Los Ãngel	70		3	1					Calle de la	TRUE	Calle de la del Manojo de Rosa			4	FALSE		Neighborh	sale	666		FALSE	129900	1856
2	21740	Piso en ve	San AndrÃ	94	54	2	2					Calle del T	FALSE	Calle del T	68		1	FALSE		Neighborh	sale	722		FALSE	144247	1535
3	21739	Piso en ve	San AndrÃ	64		2	1					Calle Pedr	TRUE	Calle Pedro JimÃ©nez			Bajo	TRUE		Neighborh	sale	583		FALSE	109900	1717
4	21738	Piso en ve	Los Rosal	108	90	2	2					Carretera	TRUE	Carretera de Villaverde a Valle			4	FALSE		Neighborh	sale	1094		FALSE	260000	2407
5	21737	Piso en ve	San AndrÃ	126	114	4	2					geologia	TRUE	geologia			3	FALSE		Neighborh	sale	901		FALSE	195000	1548
6	21736	Piso en ve	San AndrÃ	120	100	5	2					Avenida R	TRUE	Avenida Real de Pinto			1	FALSE		Neighborh	sale	884		FALSE	190000	1583
7	21735	Piso en ve	Villaverde	125	100	3	2						TRUE				2	FALSE		Neighborh	sale	912		FALSE	198500	1588
8	21734	Piso en ve	Villaverde	84	70	3	2						TRUE							Neighborh	sale	954		FALSE	212000	2524
9	21733	Piso en ve	Los Rosal	85		2	1					Calle de M	TRUE	Calle de Martinez Oviol			7	FALSE		Neighborh	sale	672		FALSE	131400	1546
10	21732	Piso en ve	San AndrÃ	69		2	2					De la Plat	TRUE	De la Plata - Villaverde			2	FALSE		Neighborh	sale	617		FALSE	118000	1710
11	21731	Piso en ve	Villaverde	122	98	3	2						TRUE				6	FALSE		Neighborh	sale	1058		FALSE	246900	2024

is_buy_pri	house_typ	is_renewa	is_new_de	built_year	has_centr	has_indivi	are_pets_	has_ac	has_fitted	has_lift	is_exterio	has_garde	has_pool	has_terra	has_balco	has_stora	is_furnish	is_kitchen	is_accessi	has_greer	energy_ce	has_parki	has_privat	has_publi	is_parking
TRUE	HouseTyp	FALSE	FALSE	1960				TRUE		FALSE	TRUE										D	FALSE			
TRUE	HouseTyp	TRUE	FALSE						TRUE	TRUE	TRUE			TRUE							en trÃ¡mit	FALSE			
TRUE	HouseTyp	FALSE	FALSE		FALSE	TRUE			TRUE	TRUE	TRUE					TRUE					no indicad	FALSE			
TRUE	HouseTyp	FALSE	FALSE	1955						TRUE	TRUE					TRUE			TRUE		en trÃ¡mit	FALSE			
TRUE	HouseTyp	FALSE	FALSE	2003				TRUE	TRUE	TRUE	TRUE		TRUE			TRUE				TRUE	en trÃ¡mit	TRUE			TRUE
TRUE	HouseTyp	FALSE	FALSE	1981	FALSE	TRUE				FALSE	TRUE			TRUE	TRUE				TRUE		en trÃ¡mit	TRUE			TRUE
TRUE	HouseTyp	FALSE	FALSE		FALSE	TRUE		TRUE	TRUE	FALSE	TRUE		TRUE	TRUE	TRUE	TRUE				TRUE	F	TRUE			TRUE

Figure 3.2 – Beginning of the Madrid Real Estate Dataset

- Preprocessing:
  Logarithmic transformation is applied to the property price to stabilize the variance and normalize the distribution.

```
class RealEstateMarketPrediction:
    def __init__(self):
        self.data = pd.read_csv("houses_Madrid.csv")
        self.model = None
        # Applying logarithmic transformation to buy price
        self.data['log_buy_price'] = np.log1p(self.data['buy_price'])
```

In the preprocessing part, features that are not useful are removed from training data. Then numerical and categorical columns are selected for transformations (Standard Scale and One Hot Encoding). Imputing the missing values in the numerical columns with median and making standardization. After that TruncatedSVD is applied for dimensional reduction in preprocessed data. It is made for reducing the overfitting.

```
    def preprocess_data(self):
        drop_features = ['Unnamed: 0', 'id', 'title', 'latitude', 'longitude', 'portal', 'door', 'rent_price_by_area',
'buy_price']
        num_columns = self.data.select_dtypes(include=['int64', 'float64']).columns.tolist()
        categorical_columns = self.data.select_dtypes(include=['object', 'bool']).columns.tolist()

        num_columns = [column for column in num_columns if column not in drop_features + ['log_buy_price']]
        categorical_columns = [column for column in categorical_columns if column not in drop_features]

        numeric_transform = Pipeline(steps=[('imputer', SimpleImputer(strategy='median')),
                                            ('scaler', StandardScaler())])
        categorical_transform = Pipeline(steps=[('imputer', SimpleImputer(strategy='most_frequent')),
                                                ('onehot', OneHotEncoder(handle_unknown='ignore'))])
        preprocess = ColumnTransformer(transformers=[('num', numeric_transform, num_columns),
                                                     ('cat', categorical_transform, categorical_columns)])

        dataset_features = self.data.drop(columns=drop_features)
        preprocessed_dataset = preprocess.fit_transform(dataset_features)

        svd = TruncatedSVD(n_components=100)
        preprocessed_dataset = svd.fit_transform(preprocessed_dataset)

        return preprocessed_dataset
```

4. *Model and Training*

Sequential model is selected with Dense layer sequence. L2 regularization is used to avoid overfitting. This regularization penalizes the sum of the squared weights and keep them small. So, it simplifies the model. Also, dropout is used for regularization. It sets a fraction of the output features of the layer to zero by selecting them randomly. It also prevents overfitting because neural network becomes less sensitive to some specific weights of the neurons.

Then model is complied with the adam optimizer and mean squared error is set as loss function.

Model train function creates a model by using the previous defined function then trains the model. Early stopping callback is used if model starts overfitting. Early stopping is controlled by monitoring the loss value. If there is not improvement in the loss value this function stops the training process. Also, a part of the data is used as validation (20%) as shown in the "validation_split".

```
def create_sequential_model(self, input):
    model = Sequential([Input(shape=(input,)),Dense(64, activation='relu', kernel_regularizer=l2(0.001)),
                        Dropout(0.2),Dense(32, activation='relu', kernel_regularizer=l2(0.001)),Dense(1)])
    model.compile(optimizer='adam', loss='mean_squared_error')
    return model

def model_train(self, x_train, y_train):
    self.model = self.create_sequential_model(x_train.shape[1])
    early_stopping = EarlyStopping(monitor='val_loss', patience=10)
    self.model.fit(x_train, y_train, batch_size=64, callbacks=[early_stopping],validation_split=0.2, epochs=100)

def model_evaluate(self, x_test, y_test):
    loss = self.model.evaluate(x_test, y_test)
    print(f"Test Loss: {loss}")
```

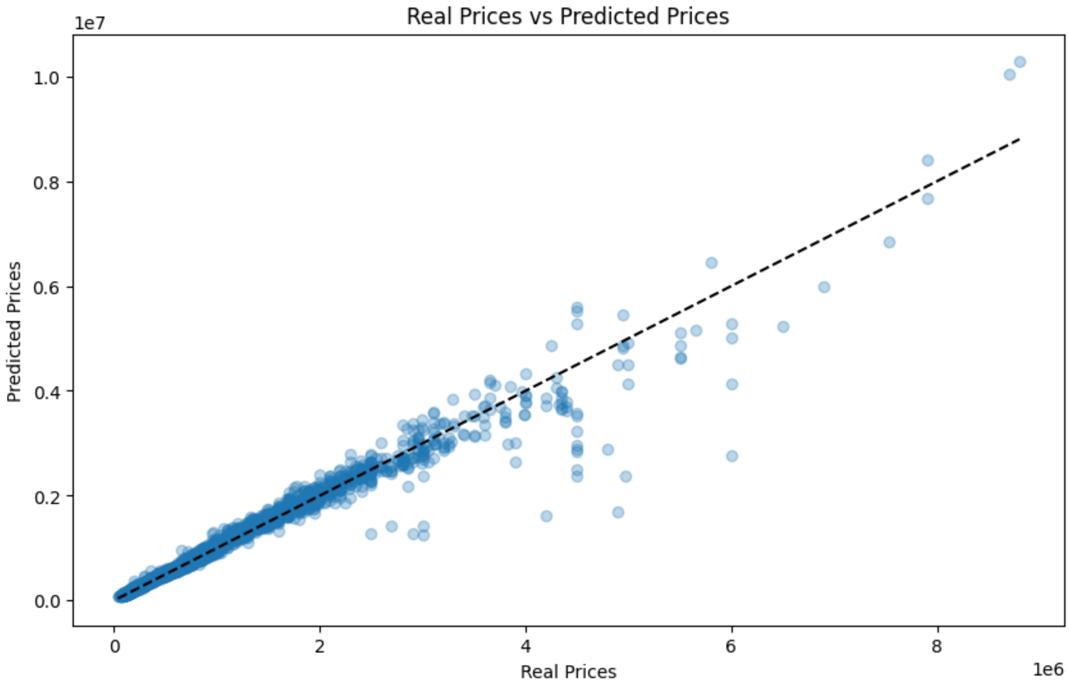
5. Results

Code given in Figure 5.1 uses RealEstateMarketPrediction class to run the model.

```
predictor = RealEstateMarketPrediction()

preprocessed_data = predictor.preprocess_data()
prediction_object = predictor.data['log_buy_price']

X_train, X_test, y_train, y_test = train_test_split(preprocessed_data, prediction_object, test_size=0.25, random_state=6)

predictor.model_train(X_train, y_train)
predictor.model_evaluate(X_test, y_test)
predictor.plot_predictions_real(X_test, y_test)
predictions = predictor.predict(X_test)

# Reverting log transformation for both predicted and real property prices
actual_prices = np.expm1(y_test)
predicted_prices = np.expm1(predictions)

for i in range(10):
    print(f"Predicted price: {round(predicted_prices[i][0])}, Actual price: {round(actual_prices.iloc[i])}")
```

Figure 5.1 – Python Code to Run the ANN Model

It can be observed that model is predicting correct for large number of real estates and predictions are close to the real price. But model cannot predict very effectively when the property price is high.



It can be also observed from the numerical results that model predicts lower prices better but it struggles at the higher prices.

```
Predicted price: 255891, Actual price: 250000
Predicted price: 157607, Actual price: 155000
Predicted price: 323821, Actual price: 289000
Predicted price: 669983, Actual price: 635000
Predicted price: 773903, Actual price: 750000
Predicted price: 265181, Actual price: 248000
Predicted price: 306665, Actual price: 295000
Predicted price: 109483, Actual price: 94000
Predicted price: 3385499, Actual price: 3300000
Predicted price: 170359, Actual price: 159000
```

6. *Discussion*

- ***Summary***

  In this project, real estate market prediction of Madrid is made with sequential Artificial Neural Network model. Implemented model shows a relationship between real estate market price and the features that affect its price.

- ***Challenges***

  Implemented model is accurately predicting the property prices in general but it struggles to predict real estate price at high values. This can be happened due to the lack of data at the high price values and its different characteristics.

  Some features have high number of missing values and these missing values are affecting the performance of the system.

- ***Future Work***

  Future research could be done to improve the model's performance on the higher price predictions. This can be done by collecting more data on the more luxury real estate and selecting different features to understand its behaviors.

  Different imputation methods could be applied to the missing values of some features. This can bias the data more effectively.

  Economic changes effect real estate market also. Features like tax, policy changes, economic conditions of the country or city should be added to the dataset for more accurate prediction in real time.

  If the range of the data could be extended over years with the added features above, real estate prediction of future months or years could be done.

7. *References*

[1] Toktogaraev, Mirbektok. (2020). Madrid Real Estate Market [Data set] Kaggle. Available at: https://www.kaggle.com/datasets/mirbektoktogaraev/madrid-real-estate-market