MCTA 4362: Machine Learning
Assignment 1

Name: Muhamad Nurhakimie Thaqif Bin Abdullah Matric No: 2213217

Answer ALL Questions.

Question 1 (20 Marks)

Please refer to the data set attached which was obtained from a census done throughout various locations in California. The dataset contains the latitude and longitude of a particular suburb/region/location, how old the houses in that area are (housing median age), total number of rooms in that location, total number of bedrooms, the population, number of families (households), the median income of the families (median income), the proximity to the ocean (Near Bay, 1 hour away,etc) and the median house value in that area. Your objective is to develop regression models to approximate the median house value in that area (dependent variables) based on the independent variables that are available

A) Based on this data set select which parameters (independent variables) will you feel will be suitable/useful to form your matrix of features. Justify your answer.
B) Perform the necessary data preparation for applying supervised machine learning algorithms, describe what data preparation was necessary (approximate missing data, removing outliers, One Hot Encoding, scaling, etc.)
C) Perform Multiple Linear Regression on this data. Evaluate and describe your model.
D) Perform Polynomial Linear Regression on this data using an order for the polynomial features of your choice. Evaluate and describe your model.
E) Perform Support Vector Regression on this data. Evaluate and describe your model.
F) Based on your results, determine the best model for this data and justify your answer.

Solution:

A) For Matrix of feature, it's better to take all the dependent variables as some variable relies on other so it either choose all or choose part of it. In the housing dataset total rooms, total bedroom, household and population can be considered as one whole matrix of feature. Leaving one of the datasets could possibly affect the model performance. In my Implementation I choose all the matrix of feature to make the coding easier.

B) Importing the data set and set the feature to be all the header except for median house value as it would be the target

Importing the dataset & Check Data

```
dataset = pd.read_csv('housing.csv')

X = dataset.drop(columns=['median_house_value']).values     # features
y = dataset['median_house_value'].values                    # target
```

```
print(X)
```

```
y = y.reshape(len(y), 1) # type: ignore

print(y)
```

For Polynomial Regression we need to convert target into NumPy array

```
y = np.array(y) #convert y to numpy array
```

There is missing value for total rooms,

```
missing_values = dataset.isnull().sum() # count of missing values in each column
print(missing_values)
```

[5] ✓ 0.0s Open 'missing_values' in Data Wrangler

```
longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
population              0
households              0
median_income           0
median_house_value      0
ocean_proximity         0
dtype: int64
```

We are using median to fill all the missing value in my case,

```
# fill missing values with the mean value of the column (Total Bedrooms)

dataset['total_bedrooms'] = dataset['total_bedrooms'].fillna(dataset['total_bedrooms'].median())


missing_values = dataset.isnull().sum() # count of missing values in each column
print(missing_values)
```

[6] ✓ 0.0s Open 'missing_values' in Data Wrangler

```
longitude           0
latitude            0
housing_median_age  0
total_rooms         0
total_bedrooms      0
population          0
households          0
median_income       0
median_house_value  0
ocean_proximity     0
dtype: int64
```

For categorical data, we apply one hot encoder to index 8 which is ocean proximity.

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder

# Handle NaN for the categorical column before encoding
dataset.iloc[:, 8] = dataset.iloc[:, 8].fillna('Unknown')

# OneHotEncoding
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [8])], remainder='passthrough')
X = np.array(ct.fit_transform(X))

# Fill any remaining NaN after encoding

# The previous fill with median was for the 'total_bedrooms' column in the original dataset.
# After OneHotEncoding, the data is transformed into a sparse matrix, and any missing values in the encoded data need to be handled separately.
# Filling with 0 ensures no NaN values remain in the transformed feature matrix.
X = pd.DataFrame(X).fillna(0).values
```

Addition For SVR is that it must had feature scaling as well as one hot encoding for categorical feature,

Feature Scaling & One Hot Encoding

```
# Handle NaN for the categorical column before encoding
dataset.iloc[:, 8] = dataset.iloc[:, 8].fillna('Unknown')

cat_index = [8]    # Ocean Proximity is in column 8
num_index = [i for i in range(X.shape[1]) if i != 8]
```

9]

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder

ct = ColumnTransformer([
    ('num_scaler', StandardScaler(), num_index),   # Scale numerical features
    ('cat', OneHotEncoder(), cat_index)            # OneHotEncode categorical feature
])
```

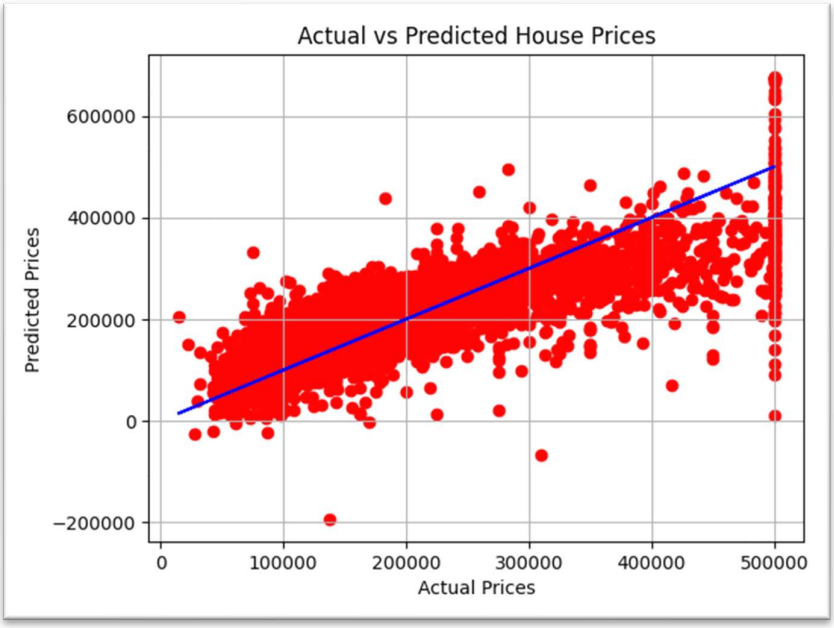
]

```
X = ct.fit_transform(X)
sc_y = StandardScaler()
y = sc_y.fit_transform(y)
```

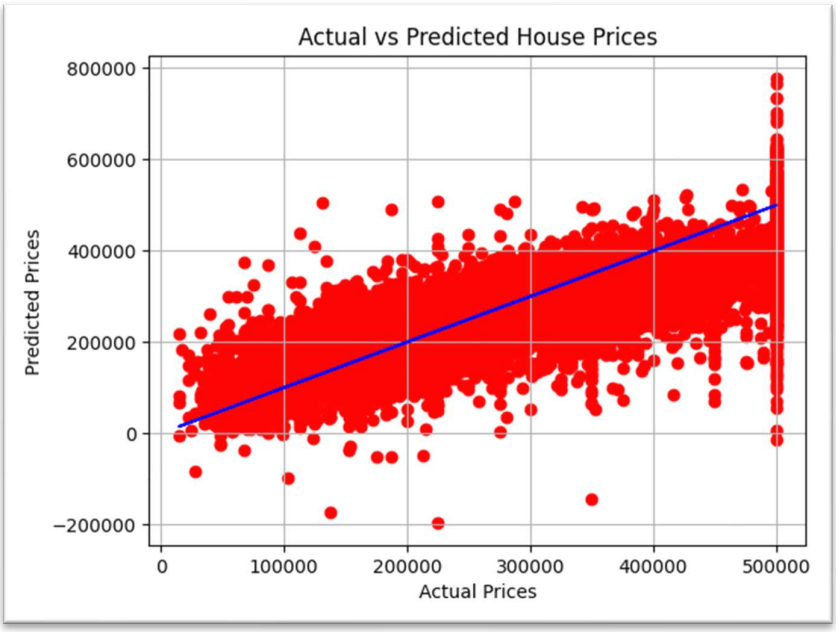
]

```
X = pd.DataFrame(X).fillna(0).values # fill NaN values with 0 for after scaling
```

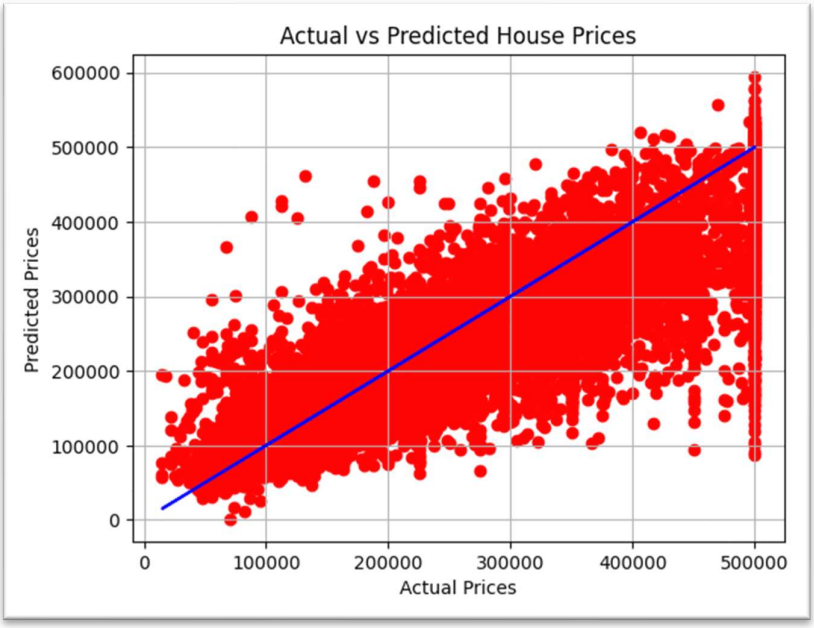
C) **R-squared**: 0.6368, **MAE**: 49867.71, **RMSE**: 68814.66



D) **R-squared**: 0.7103, **MAE**: 44215.20, **RMSE**: 62113.01



E) **R-squared**: 0.7757, **MAE**: 35600.33, **RMSE**: 54647.24



F) Overall, Support Vector Regression (SVR) perform the best follow by polynomial Linear Regression (PLR) and Multiple Linear Regression (MLR) as the overall R-squared accuracy is the highest (Most Predictive Power), Lowest Mean Absolute Error (MAE) Meaning it had smaller error compared to other model and lowest Root Mean Squared Error (RMSE) (Less Extreme Error)

Question 2 (20 Marks)

Solar-based energy is becoming one of the most promising sources for producing power for residential, commercial, and industrial applications. Energy production based on solar photovoltaic (PV) systems has gained much attention from researchers and practitioners recently due to its desirable characteristics. However, the main difficulty in solar energy production is the volatility intermittent of photovoltaic system power generation, which is mainly due to weather conditions. For the large-scale solar farms, the power imbalance of the photovoltaic system may cause a significant loss in their economic profit. Accurate forecasting of the power output of PV systems in a short term is of great importance for daily/hourly efficient management of power grid production, delivery, and storage, as well as for decision-making on the energy market, facilitating early participation in energy auction markets and efficient resource planning.

A) Based on this data set select which parameters (independent variables) will you feel will be suitable/useful to form your matrix of features. Justify your answer.

B) Perform the necessary data preparation for applying supervised machine learning algorithms, describe what data preparation was necessary (approximate missing data, removing outliers, One Hot Encoding, scaling, etc.)

C) Perform Multiple Linear Regression on this data. Evaluate and describe your model.

D) Perform Polynomial Linear Regression on this data using an order for the polynomial features of your choice. Evaluate and describe your model.

E) Perform Support Vector Regression on this data. Evaluate and describe your model.

F) Based on your results, determine the best model for this data and justify your answer.

Solution:

A) For this dataset I decide also to use all the features as some features depending on other this would also improve the model and make it more robust as it considers of all possibility

B) The data given is in excel format, so I convert it first into .csv file for easier visualization

```
import pandas as pd

# Load the Excel file
excel_data = pd.read_excel('Solar_Power_Generation.xlsx')

# Save as CSV
excel_data.to_csv('Solar_Power_Generation.csv', index=False)
```

Import the dataset, and choose all the feature except for the last index would be generated power which would be the target

Importing the dataset & Check Data

```
dataset = pd.read_csv('Solar_Power_Generation.csv')

X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values


print(X)


y = y.reshape(len(y), 1) # type: ignore

print(y)
```

For Polynomial Regression we need to convert it into NumPy array

```
y = np.array(y) #convert y to numpy array
```

Check for missing value,

```
missing_values = dataset.isnull().sum() # count of missing values in each column
print(missing_values)
```

[5]

```
temperature_2_m_above_gnd           1
relative_humidity_2_m_above_gnd     0
mean_sea_level_pressure_MSL         0
total_precipitation_sfc             0
snowfall_amount_sfc                 0
total_cloud_cover_sfc               0
high_cloud_cover_high_cld_lay       0
medium_cloud_cover_mid_cld_lay      0
low_cloud_cover_low_cld_lay         0
shortwave_radiation_backwards_sfc   1
wind_speed_10_m_above_gnd           0
wind_direction_10_m_above_gnd       1
wind_speed_80_m_above_gnd           0
wind_direction_80_m_above_gnd       0
wind_speed_900_mb                   0
wind_direction_900_mb               0
wind_gust_10_m_above_gnd            0
angle_of_incidence                  0
zenith                              0
azimuth                             0
generated_power_kw                  1
dtype: int64
```

Then, using Median to fill out missing value

```
# fill missing values with the mean value of the column (Total Bedrooms)

dataset = dataset.fillna(dataset.median())

missing_values = dataset.isnull().sum() # count of missing values in each column
print(missing_values)
```

[6]

```
temperature_2_m_above_gnd           0
relative_humidity_2_m_above_gnd     0
mean_sea_level_pressure_MSL         0
total_precipitation_sfc             0
snowfall_amount_sfc                 0
total_cloud_cover_sfc               0
high_cloud_cover_high_cld_lay       0
medium_cloud_cover_mid_cld_lay      0
low_cloud_cover_low_cld_lay         0
shortwave_radiation_backwards_sfc   0
wind_speed_10_m_above_gnd           0
wind_direction_10_m_above_gnd       0
wind_speed_80_m_above_gnd           0
wind_direction_80_m_above_gnd       0
wind_speed_900_mb                   0
wind_direction_900_mb               0
wind_gust_10_m_above_gnd            0
angle_of_incidence                  0
zenith                              0
azimuth                             0
generated_power_kw                  0
dtype: int64
```

Making sure the value is in the data frame as well as data set to prevent any error related to Nan Missing.

```
# fill missing values with the median value

X = pd.DataFrame(X).fillna(pd.DataFrame(X).median()).values
y = pd.DataFrame(y).fillna(pd.DataFrame(y).median()).values
```

[10]

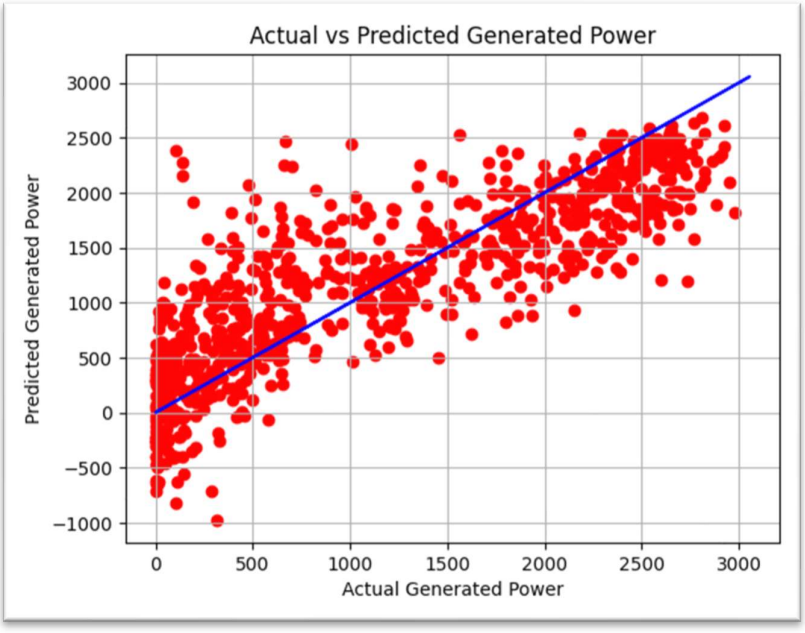
For Support Vector Regression we need to had feature scaling

Feature Scaling

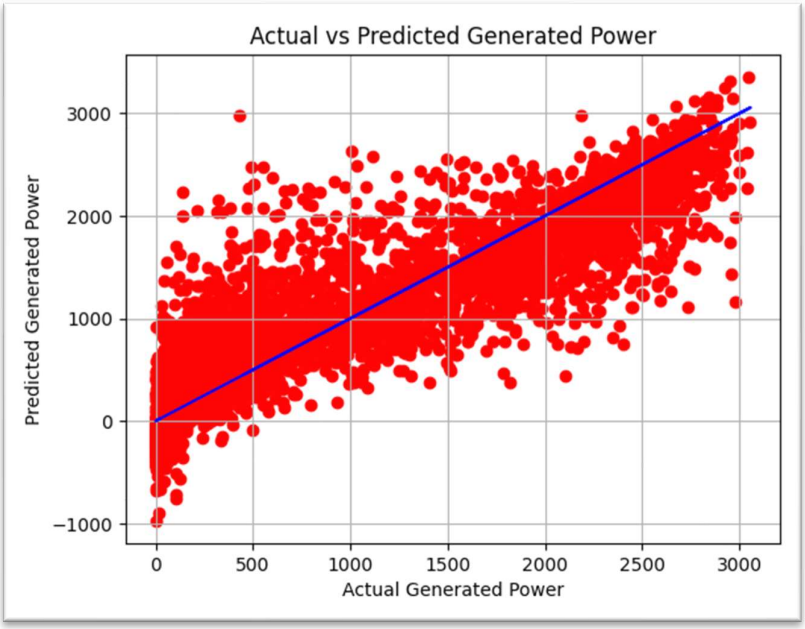
```
from sklearn.preprocessing import StandardScaler

# Standard Scaler = x - mean(x) / standard deviation(x)
# Range of standard scaler is -3 to 3

sc_X = StandardScaler()
sc_y = StandardScaler()
X = sc_X.fit_transform(X)
y = sc_y.fit_transform(y)
```

[13]

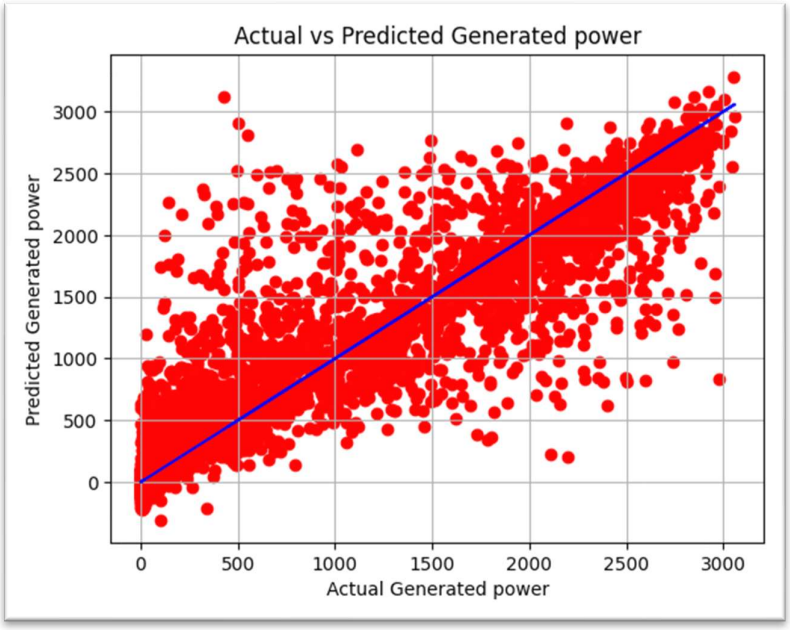
C) **R-squared**: 0.6966, **MAE**: 395.64, **RMSE**: 515.21



D) **R-squared**: 0.7943, **MAE**: 308.00, **RMSE**: 425.35



E) **R-squared**: 0.8271, **MAE**: 239.06, **RMSE**: 389.96



F) Overall, for the solar dataset we can see that support vector regression perform the best follow by polynomials regression and multiple linear regression as the overall R-squared accuracy is the highest (Most Predictive Power), Lowest Mean Absolute Error (MAE) Meaning it had smaller error compared to other model and lowest Root Mean Squared Error (RMSE) (Less Extreme Error)