21102b0060-ml-exp5

September 25, 2024

Vedika Desai
21102B0060
BE CMPN B
ML assignment 5
github:https://github.com/desaivedika/ML-Exp_Sem-7.git

[2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import  seaborn as sns
```

[3]:
```
df=pd.read_csv('bc_ml.csv')
```

[4]:
```
df
```

[4]:

	ID	Diagnosis	radius1	texture1	perimeter1	area1	smoothness1	\
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	
..	…	…	…	…	…	…	…	
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	
566	926954	M	16.60	28.08	108.30	858.1	0.08455	
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	
568	92751	B	7.76	24.54	47.92	181.0	0.05263	

	compactness1	concavity1	concave_points1	…	radius3	texture3	\
0	0.27760	0.30010	0.14710	…	25.380	17.33	
1	0.07864	0.08690	0.07017	…	24.990	23.41	
2	0.15990	0.19740	0.12790	…	23.570	25.53	
3	0.28390	0.24140	0.10520	…	14.910	26.50	
4	0.13280	0.19800	0.10430	…	22.540	16.67	
..	…	…	…	…	…	…	
564	0.11590	0.24390	0.13890	…	25.450	26.40	
565	0.10340	0.14400	0.09791	…	23.690	38.25	
566	0.10230	0.09251	0.05302	…	18.980	34.12	

```
567       0.27700     0.35140          0.15200  …   25.740     39.42
568       0.04362     0.00000          0.00000  …    9.456     30.37

     perimeter3   area3  smoothness3  compactness3  concavity3  \
0         184.60  2019.0      0.16220       0.66560      0.7119
1         158.80  1956.0      0.12380       0.18660      0.2416
2         152.50  1709.0      0.14440       0.42450      0.4504
3          98.87   567.7      0.20980       0.86630      0.6869
4         152.20  1575.0      0.13740       0.20500      0.4000
..           …     …           …             …          …
564       166.10  2027.0      0.14100       0.21130      0.4107
565       155.00  1731.0      0.11660       0.19220      0.3215
566       126.70  1124.0      0.11390       0.30940      0.3403
567       184.60  1821.0      0.16500       0.86810      0.9387
568        59.16   268.6      0.08996       0.06444      0.0000

     concave_points3  symmetry3  fractal_dimension3
0             0.2654     0.4601             0.11890
1             0.1860     0.2750             0.08902
2             0.2430     0.3613             0.08758
3             0.2575     0.6638             0.17300
4             0.1625     0.2364             0.07678
..               …          …                  …
564           0.2216     0.2060             0.07115
565           0.1628     0.2572             0.06637
566           0.1418     0.2218             0.07820
567           0.2650     0.4087             0.12400
568           0.0000     0.2871             0.07039

[569 rows x 32 columns]
```

[5]:
```
df.isnull().sum()
```

[5]:
```
ID                   0
Diagnosis            0
radius1              0
texture1             0
perimeter1           0
area1                0
smoothness1          0
compactness1         0
concavity1           0
concave_points1      0
symmetry1            0
fractal_dimension1   0
radius2              0
texture2             0
```

```
perimeter2            0
area2                 0
smoothness2           0
compactness2          0
concavity2            0
concave_points2       0
symmetry2             0
fractal_dimension2    0
radius3               0
texture3              0
perimeter3            0
area3                 0
smoothness3           0
compactness3          0
concavity3            0
concave_points3       0
symmetry3             0
fractal_dimension3    0
dtype: int64
```

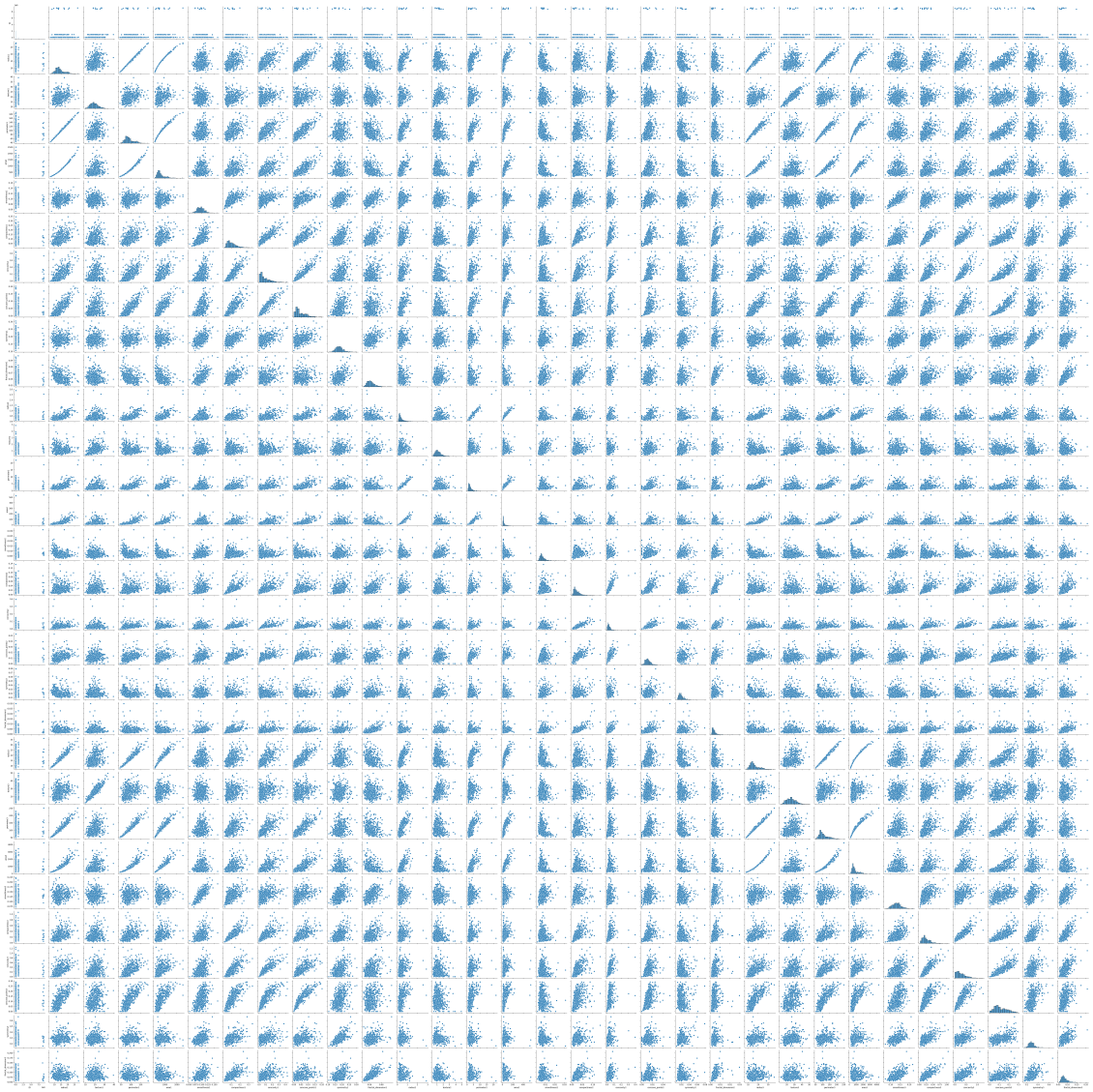
[6]:
```
df.dtypes
```

[6]:
```
ID                      int64
Diagnosis              object
radius1               float64
texture1              float64
perimeter1            float64
area1                 float64
smoothness1           float64
compactness1          float64
concavity1            float64
concave_points1       float64
symmetry1             float64
fractal_dimension1    float64
radius2               float64
texture2              float64
perimeter2            float64
area2                 float64
smoothness2           float64
compactness2          float64
concavity2            float64
concave_points2       float64
symmetry2             float64
fractal_dimension2    float64
radius3               float64
texture3              float64
perimeter3            float64
```

```
area3                 float64
smoothness3           float64
compactness3          float64
concavity3            float64
concave_points3       float64
symmetry3             float64
fractal_dimension3    float64
dtype: object
```

[8]:
```
sns.pairplot(df)
```

[8]:
```
<seaborn.axisgrid.PairGrid at 0x78e2cb9b9180>
```



[9]:
```
df.describe()
```

[9]:

	ID	radius1	texture1	perimeter1	area1	\
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	

	smoothness1	compactness1	concavity1	concave_points1	symmetry1	\
count	569.000000	569.000000	569.000000	569.000000	569.000000	
mean	0.096360	0.104341	0.088799	0.048919	0.181162	
std	0.014064	0.052813	0.079720	0.038803	0.027414	
min	0.052630	0.019380	0.000000	0.000000	0.106000	
25%	0.086370	0.064920	0.029560	0.020310	0.161900	
50%	0.095870	0.092630	0.061540	0.033500	0.179200	
75%	0.105300	0.130400	0.130700	0.074000	0.195700	
max	0.163400	0.345400	0.426800	0.201200	0.304000	

	…	radius3	texture3	perimeter3	area3	smoothness3	\
count	…	569.000000	569.000000	569.000000	569.000000	569.000000	
mean	…	16.269190	25.677223	107.261213	880.583128	0.132369	
std	…	4.833242	6.146258	33.602542	569.356993	0.022832	
min	…	7.930000	12.020000	50.410000	185.200000	0.071170	
25%	…	13.010000	21.080000	84.110000	515.300000	0.116600	
50%	…	14.970000	25.410000	97.660000	686.500000	0.131300	
75%	…	18.790000	29.720000	125.400000	1084.000000	0.146000	
max	…	36.040000	49.540000	251.200000	4254.000000	0.222600	

	compactness3	concavity3	concave_points3	symmetry3	\
count	569.000000	569.000000	569.000000	569.000000	
mean	0.254265	0.272188	0.114606	0.290076	
std	0.157336	0.208624	0.065732	0.061867	
min	0.027290	0.000000	0.000000	0.156500	
25%	0.147200	0.114500	0.064930	0.250400	
50%	0.211900	0.226700	0.099930	0.282200	
75%	0.339100	0.382900	0.161400	0.317900	
max	1.058000	1.252000	0.291000	0.663800	

	fractal_dimension3
count	569.000000
mean	0.083946
std	0.018061
min	0.055040

```
25%              0.071460
50%              0.080040
75%              0.092080
max              0.207500

[8 rows x 31 columns]
```

[10]:
```
df['Diagnosis'] = df['Diagnosis'].replace({'B': 0, 'M': 1})
```

[11]:
```
X=df.drop("Diagnosis",axis=1)
y=df['Diagnosis']
```

[12]:
```
print(y)
```

```
0      1
1      1
2      1
3      1
4      1
      ..
564    1
565    1
566    1
567    1
568    0
Name: Diagnosis, Length: 569, dtype: int64
```

[13]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
```

[14]:
```
import xgboost as xgb
```

[15]:
```
model = xgb.XGBClassifier()
```

[16]:
```
print(f"X_train shape: {X_train.shape}")
print(f"y_train shape: {y_train.shape}")
```

```
X_train shape: (426, 31)
y_train shape: (426,)
```

[17]:
```
model.fit(X_train, y_train)
```

[17]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
```

```
                   interaction_constraints=None, learning_rate=None, max_bin=None,
                   max_cat_threshold=None, max_cat_to_onehot=None,
                   max_delta_step=None, max_depth=None, max_leaves=None,
                   min_child_weight=None, missing=nan, monotone_constraints=None,
                   multi_strategy=None, n_estimators=None, n_jobs=None,
                   num_parallel_tree=None, random_state=None, …)
```

[18]:
```
predictions = model.predict(X_test)
```

[19]:
```
from sklearn.metrics import accuracy_score,precision_score, recall_score,␣
↪f1_score, roc_auc_score, roc_curve
```

[20]:
```
accuracy = accuracy_score(y_test, predictions)
```

[21]:
```
print(accuracy)
```

```
0.972027972027972
```

[22]:
```
precision = precision_score(y_test, predictions)
recall = recall_score(y_test, predictions)
f1 = f1_score(y_test, predictions)
```

[23]:
```
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1:.2f}")
```

```
Precision: 0.96
Recall: 0.96
F1 Score: 0.96
```

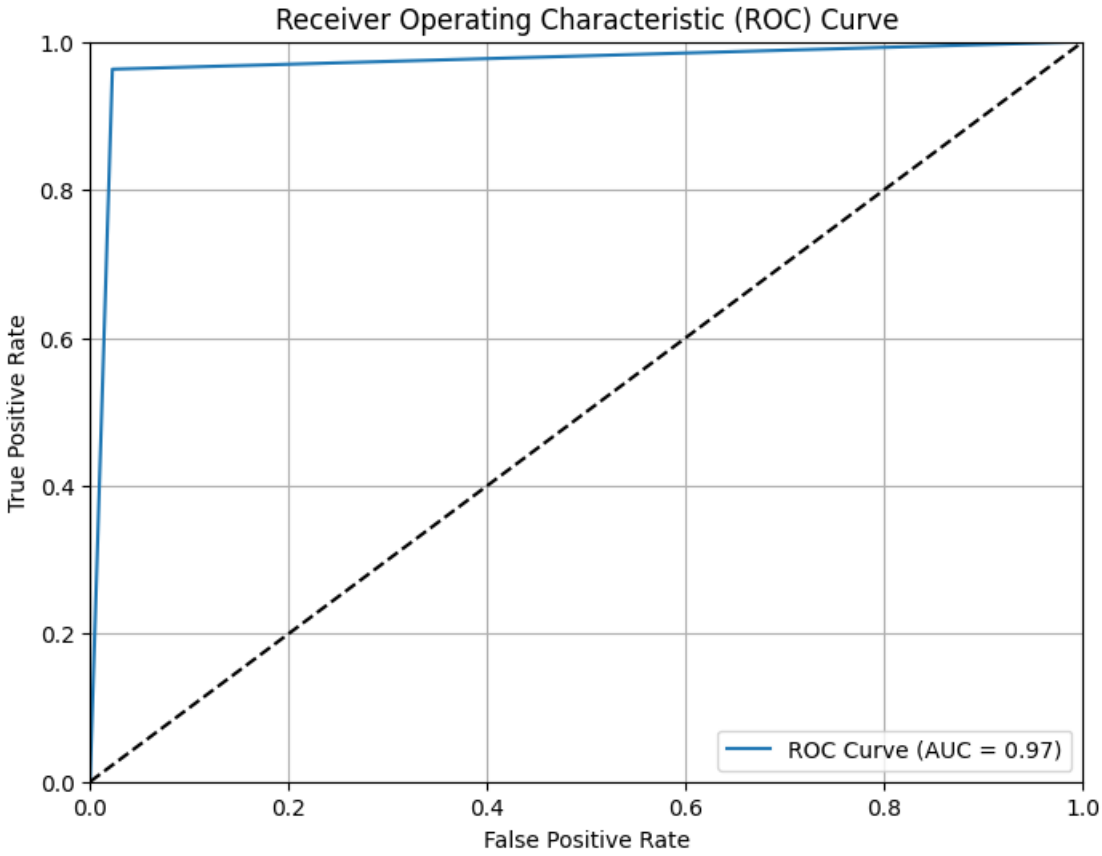
[24]:
```
roc_auc = roc_auc_score(y_test, predictions)
print(f"ROC AUC: {roc_auc:.2f}")
```

```
ROC AUC: 0.97
```

[25]:
```
fpr, tpr, thresholds = roc_curve(y_test, predictions)

# Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], 'k--')  # Diagonal line
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
```

```
plt.grid()
plt.show()
```



[26]:
```
## Regression
```

[27]:
```
df2=pd.read_csv('housing.csv')
```

[28]:
```
print(df2)
```

```
       longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0        -122.23     37.88                41.0        880.0           129.0
1        -122.22     37.86                21.0       7099.0          1106.0
2        -122.24     37.85                52.0       1467.0           190.0
3        -122.25     37.85                52.0       1274.0           235.0
4        -122.25     37.85                52.0       1627.0           280.0
...          ...       ...                 ...          ...             ...
20635    -121.09     39.48                25.0       1665.0           374.0
20636    -121.21     39.49                18.0        697.0           150.0
20637    -121.22     39.43                17.0       2254.0           485.0
20638    -121.32     39.43                18.0       1860.0           409.0
```

```
20639    -121.24     39.37                16.0       2785.0           616.0

       population  households  median_income  median_house_value  \
0           322.0       126.0         8.3252            452600.0
1          2401.0      1138.0         8.3014            358500.0
2           496.0       177.0         7.2574            352100.0
3           558.0       219.0         5.6431            341300.0
4           565.0       259.0         3.8462            342200.0
…             …           …              …                   …
20635       845.0       330.0         1.5603             78100.0
20636       356.0       114.0         2.5568             77100.0
20637      1007.0       433.0         1.7000             92300.0
20638       741.0       349.0         1.8672             84700.0
20639      1387.0       530.0         2.3886             89400.0

      ocean_proximity
0            NEAR BAY
1            NEAR BAY
2            NEAR BAY
3            NEAR BAY
4            NEAR BAY
…               …
20635          INLAND
20636          INLAND
20637          INLAND
20638          INLAND
20639          INLAND

[20640 rows x 10 columns]
```

[29]:
```
df2.isnull().sum()
```

[29]:
```
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
```

[30]:
```
df3=df2.dropna()
```

[31]:
```
df3
```

[31]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	\
0	-122.23	37.88	41.0	880.0	129.0	
1	-122.22	37.86	21.0	7099.0	1106.0	
2	-122.24	37.85	52.0	1467.0	190.0	
3	-122.25	37.85	52.0	1274.0	235.0	
4	-122.25	37.85	52.0	1627.0	280.0	
…	…	…	…	…	…	
20635	-121.09	39.48	25.0	1665.0	374.0	
20636	-121.21	39.49	18.0	697.0	150.0	
20637	-121.22	39.43	17.0	2254.0	485.0	
20638	-121.32	39.43	18.0	1860.0	409.0	
20639	-121.24	39.37	16.0	2785.0	616.0	

	population	households	median_income	median_house_value	\
0	322.0	126.0	8.3252	452600.0	
1	2401.0	1138.0	8.3014	358500.0	
2	496.0	177.0	7.2574	352100.0	
3	558.0	219.0	5.6431	341300.0	
4	565.0	259.0	3.8462	342200.0	
…	…	…	…	…	
20635	845.0	330.0	1.5603	78100.0	
20636	356.0	114.0	2.5568	77100.0	
20637	1007.0	433.0	1.7000	92300.0	
20638	741.0	349.0	1.8672	84700.0	
20639	1387.0	530.0	2.3886	89400.0	

	ocean_proximity
0	NEAR BAY
1	NEAR BAY
2	NEAR BAY
3	NEAR BAY
4	NEAR BAY
…	…
20635	INLAND
20636	INLAND
20637	INLAND
20638	INLAND
20639	INLAND

[20433 rows x 10 columns]

[32]:
```
df3.isnull().sum()
```

[32]:
```
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
```

[33]:
```
df3=df3.drop('latitude',axis=1)
df3=df3.drop('longitude',axis=1)
df3=df3.drop('ocean_proximity',axis=1)
```

[34]:
```
df3
```

[34]:

	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
0	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0
1	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0
2	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0
3	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0
4	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0
…	…	…	…	…	…	…	…
20635	25.0	1665.0	374.0	845.0	330.0	1.5603	78100.0
20636	18.0	697.0	150.0	356.0	114.0	2.5568	77100.0
20637	17.0	2254.0	485.0	1007.0	433.0	1.7000	92300.0
20638	18.0	1860.0	409.0	741.0	349.0	1.8672	84700.0
20639	16.0	2785.0	616.0	1387.0	530.0	2.3886	89400.0

[20433 rows x 7 columns]

[35]:
```
model1 = xgb.XGBRegressor()
```

[36]:
```
X1=df3.drop('median_house_value',axis=1)
y1=df3['median_house_value']
```

[37]:
```
X_train1, X_test1, y_train1, y_test1 = train_test_split(X1, y1, random_state=42)
```

[38]:
```
model1.fit(X_train1,y_train1)
```

[38]:
```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=None, …)
```

[39]:
```
y_pred= model1.predict(X_test1)
```

[40]:
```
print(y_pred)
```

```
[226892.05  135920.73  187461.75  … 172809.42  183101.03  100345.914]
```

[41]:
```
from sklearn.metrics import mean_squared_error, r2_score
```

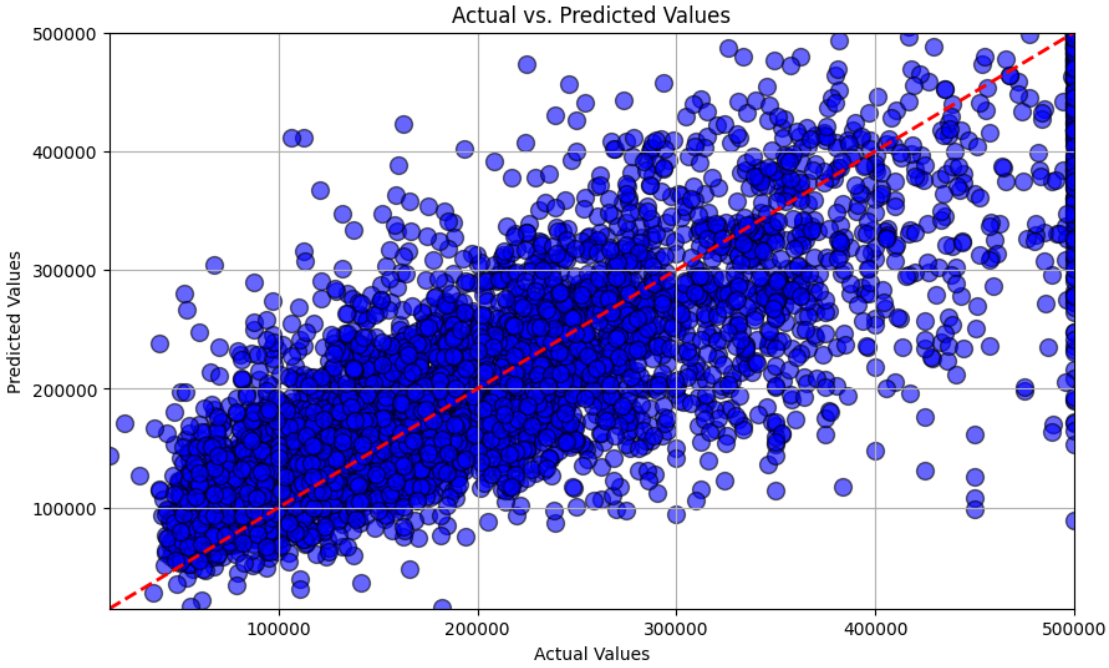
[42]:
```
mse = mean_squared_error(y_test1, y_pred)
print(f"Mean Squared Error: {mse:.2f}")
```

```
Mean Squared Error: 4528594728.87
```

[43]:
```
r2 = r2_score(y_test1, y_pred)
print(f"R² Score: {r2:.2f}")
```

```
R² Score: 0.67
```

[44]:
```
plt.figure(figsize=(10, 6))
plt.scatter(y_test1, y_pred, color='blue', alpha=0.6, edgecolors='k', s=100)
plt.plot([y_test1.min(), y_test1.max()], [y_test1.min(), y_test1.max()], 'r--',␣
 ↪lw=2)
plt.title('Actual vs. Predicted Values')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.xlim([y_test1.min(), y_test1.max()])
plt.ylim([y_test1.min(), y_test1.max()])
plt.grid()
plt.show()
```


Actual vs. Predicted Values
Predicted Values
Actual Values
500000
400000
300000
200000
100000
100000
200000
300000
400000
500000