Лабораторная работа №3

Выполнил студент группы БВТ2003 Глазков Даниил

In [23]:
```
import numpy as np
from matplotlib import pyplot as plt
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.datasets import boston_housing
(train_data, train_targets), (test_data, test_targets) = boston_housing.load_dat
print(train_data.shape)
print(test_data.shape)
print(test_targets)
```

```
(404, 13)
(102, 13)
[ 7.2 18.8 19.  27.  22.2 24.5 31.2 22.9 20.5 23.2 18.6 14.5 17.8 50.
 20.8 24.3 24.2 19.8 19.1 22.7 12.  10.2 20.  18.5 20.9 23.  27.5 30.1
  9.5 22.  21.2 14.1 33.1 23.4 20.1  7.4 15.4 23.8 20.1 24.5 33.  28.4
 14.1 46.7 32.5 29.6 28.4 19.8 20.2 25.  35.4 20.3  9.7 14.5 34.9 26.6
  7.2 50.  32.4 21.6 29.8 13.1 27.5 21.2 23.1 21.9 13.  23.2  8.1  5.6
 21.7 29.6 19.6  7.  26.4 18.9 20.9 28.1 35.4 10.2 24.3 43.1 17.6 15.4
 16.2 27.1 21.4 21.5 22.4 25.  16.6 18.6 22.  42.8 35.1 21.5 36.  21.9
 24.1 50.  26.7 25. ]
```

In [24]:
```
mean = train_data.mean(axis=0)
train_data -= mean
std = train_data.std(axis=0)
```

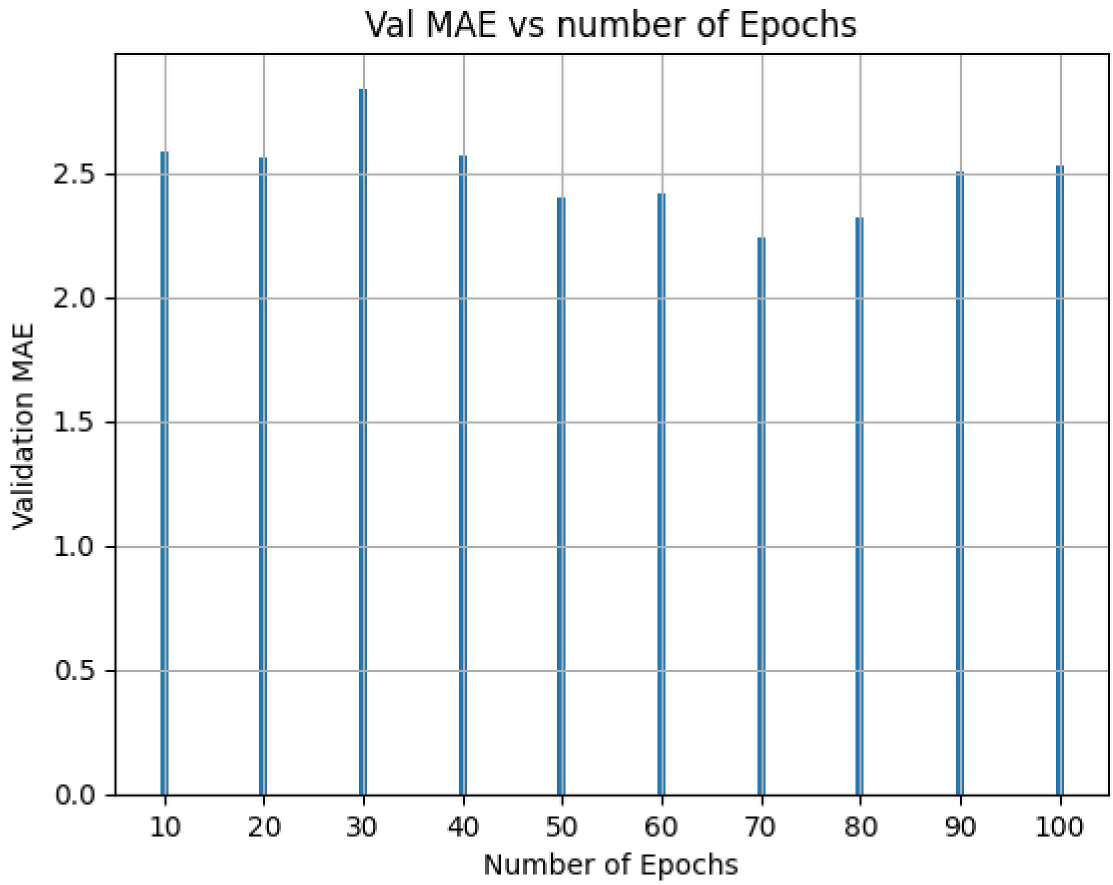
In [25]:
```
train_data /= std
test_data -= mean
test_data /= std
```

In [26]:
```
def build_model():
    model = Sequential()
    model.add(Dense(64, activation='relu', input_shape=(train_data.shape[1],)))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(1))
    model.compile(optimizer='rmsprop', loss='mse', metrics=['mae'])
    return model
```

In [49]:
```
def train_and_plot_epochs(data, targets, num_epochs_range):
    mae_history = []

    for num_epochs in num_epochs_range:
        print(f"Training model with {num_epochs} epochs...")
        model = build_model()
        history = model.fit(partial_train_data, partial_train_targets, epochs=nu
        mae_history.append(history.history['val_mae'][-1])
    return mae_history

num_epochs_range = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
mae_history = train_and_plot_epochs(train_data,train_targets, num_epochs_range)
```

```
plt.bar(num_epochs_range, mae_history)
plt.xlabel('Number of Epochs')
plt.ylabel('Validation MAE')
plt.title('Val MAE vs number of Epochs')
plt.xticks(num_epochs_range)

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

```
Training model with 10 epochs...
Training model with 20 epochs...
Training model with 30 epochs...
Training model with 40 epochs...
Training model with 50 epochs...
Training model with 60 epochs...
Training model with 70 epochs...
Training model with 80 epochs...
Training model with 90 epochs...
Training model with 100 epochs...
```

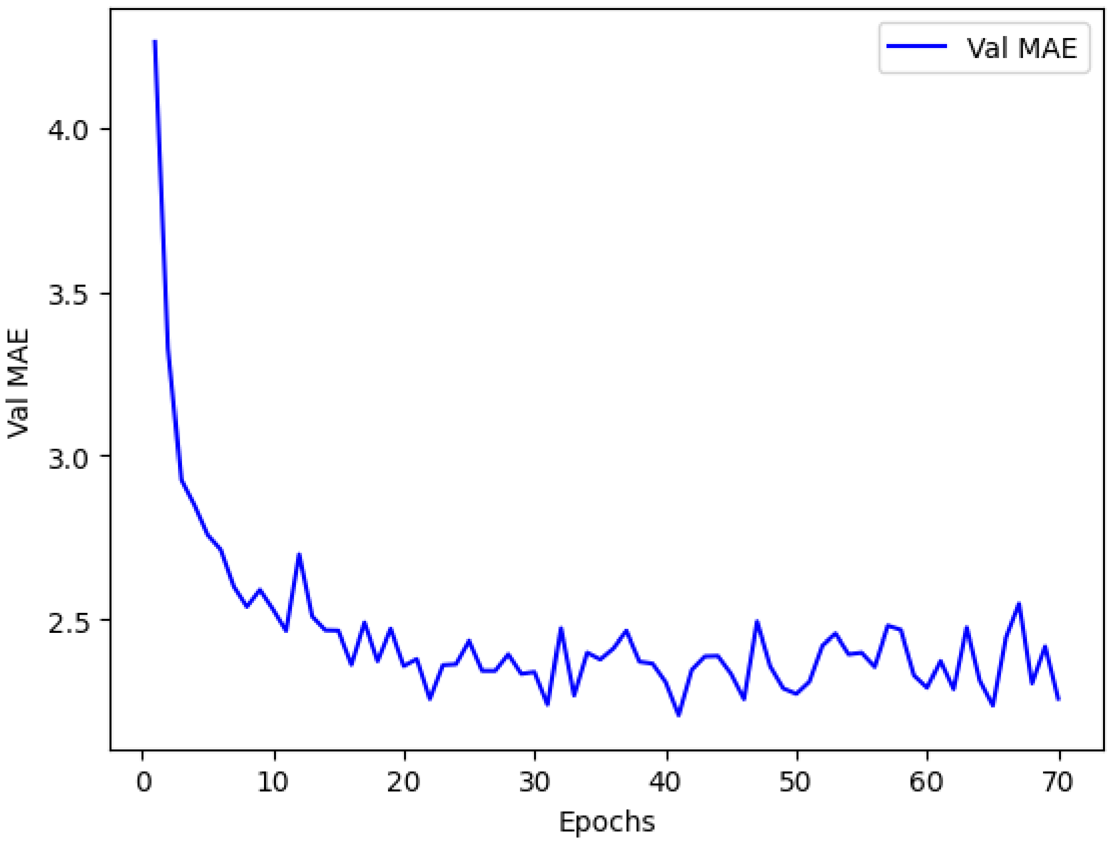


In [27]:
```
k = 4
num_val_samples = len(train_data) // k
num_epochs = 70
all_scores = []
all_mae_results = []
all_loss_results = []
for i in range(k):
    print('processing fold #', i)
    val_data = train_data[i * num_val_samples: (i + 1) * num_val_samples]
    val_targets = train_targets[i * num_val_samples: (i + 1) * num_val_samples]
    partial_train_data = np.concatenate([train_data[:i * num_val_samples], trair
    partial_train_targets = np.concatenate([train_targets[:i * num_val_samples],
    model = build_model()
```

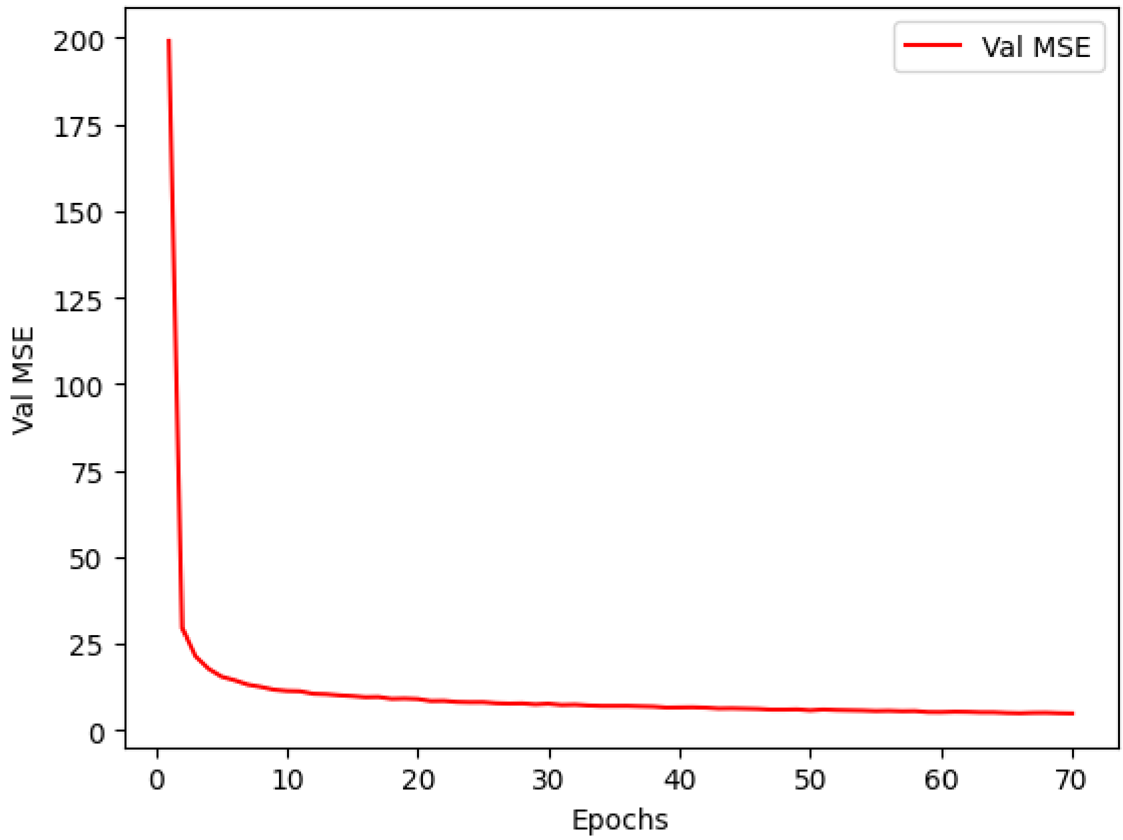
```
    model_result = model.fit(partial_train_data, partial_train_targets, epochs=r
    val_mse, val_mae = model.evaluate(val_data, val_targets, verbose=0)
    all_scores.append(val_mae)
    mae_result = model_result.history['val_mae']
    loss_result = model_result.history['loss']
    all_mae_results.append(mae_result)
    all_loss_results.append(loss_result)

print(np.mean(all_scores))
```

```
processing fold # 0
processing fold # 1
processing fold # 2
processing fold # 3
2.259992629289627
```

In [29]:
```
plt.plot(range(1, num_epochs + 1), np.mean(all_mae_results, axis = 0), label = '
plt.xlabel('Epochs')
plt.ylabel('Val MAE')
plt.legend()

plt.show()
```

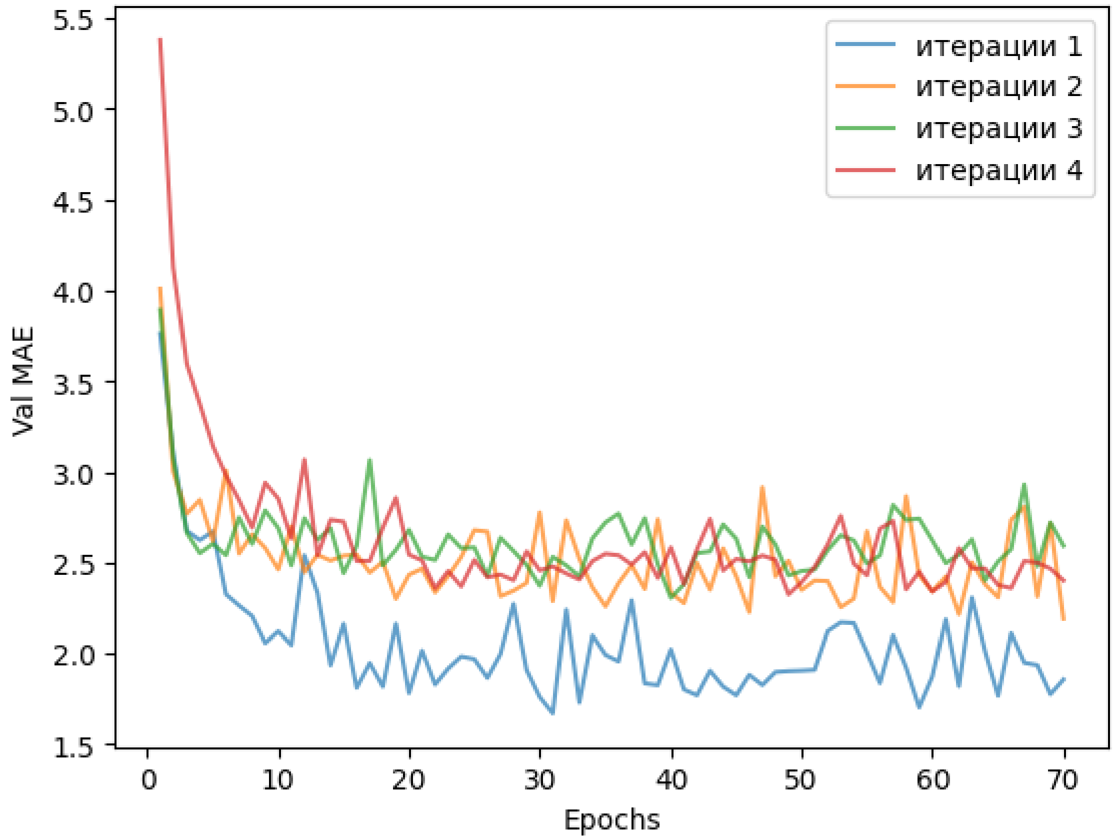


In [30]:
```
plt.plot(range(1, num_epochs + 1), np.mean(all_loss_results, axis = 0), label =
plt.xlabel('Epochs')
plt.ylabel('Val MSE')
plt.legend()

plt.show()
```



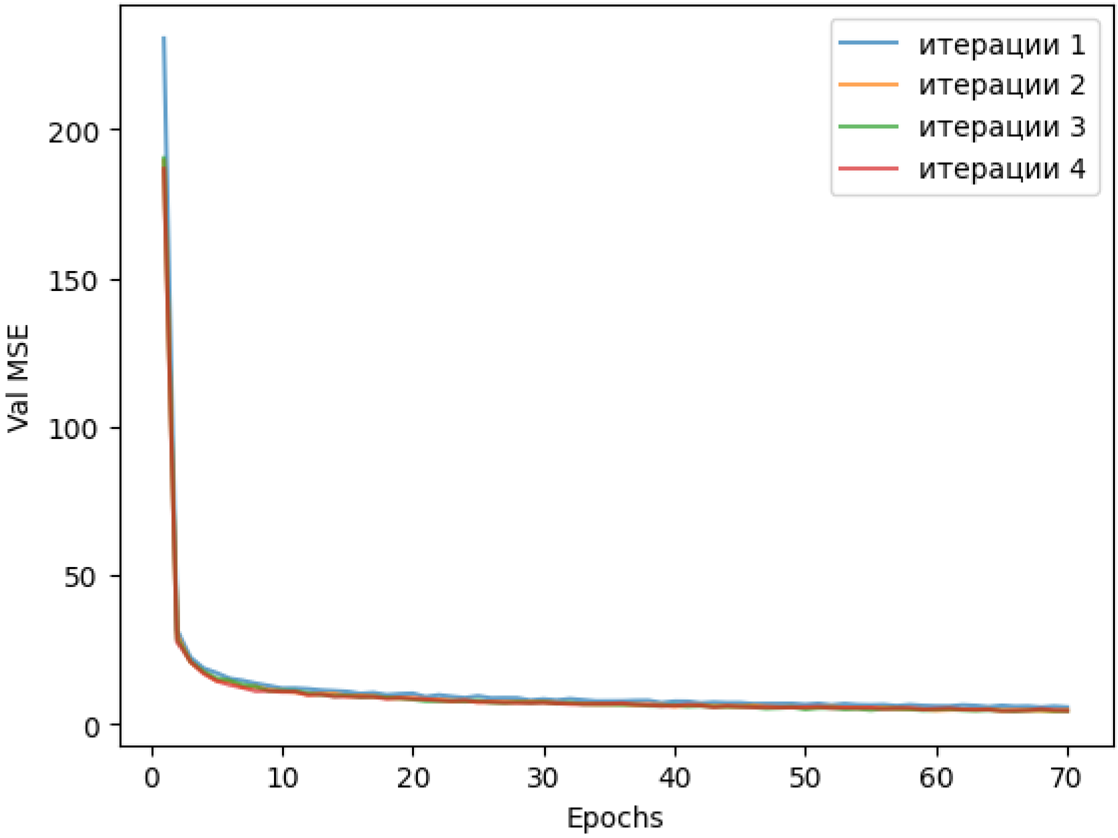
In [31]:
```
for i in range(k):
    plt.plot(range(1, num_epochs + 1), all_mae_results[i], label = f'итерации {i
plt.xlabel('Epochs')
plt.ylabel('Val MAE')
plt.legend()
```

Out[31]: <matplotlib.legend.Legend at 0x1de791a88d0>



In [32]:
```
for i in range(k):
    plt.plot(range(1, num_epochs + 1), all_loss_results[i], label = f'итерации {
plt.xlabel('Epochs')
plt.ylabel('Val MSE')
plt.legend()
```

Out[32]: `<matplotlib.legend.Legend at 0x1de793788d0>`



Вывод: мы выяснили что задачи классификации и регрессии отличаются типом целевого признака: в регрессии целевой признак количественный (числовой), в классификации - категориальный, выяснили что кол-ва эпох влияет на качество обучения и может приводить к недообучению или переобучению, применили перекрестную проверку по К блокам при различных К и построить графики ошибки и точности во время обучения для моделей, а также усредненные графики по всем моделям.